

20070125 PACER 66 1st Amd Complaint-Ex F Pt 1.pdf

**(12) United States Patent**  
**Hansen et al.**

(10) **Patent No.:** **US 6,725,356 B2**  
(45) **Date of Patent:** **\*Apr. 20, 2004**

**(54) SYSTEM WITH WIDE OPERAND ARCHITECTURE, AND METHOD**

4,701,875 A 10/1987 Konishi et al.

(List continued on next page.)

(75) Inventors: **Craig Hansen**, Los Altos, CA (US);  
**John Moussouris**, Palo Alto, CA (US)

## FOREIGN PATENT DOCUMENTS

(73) Assignee: **MicroUnity Systems Engineering, Inc., Sunnyvale, CA (US)**

DE	0 654 733 A1	5/1994
JP	0 474 246 A2	6/1991

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

## OTHER PUBLICATIONS

This patent is subject to a terminal disclaimer.

Parallel Computers for Graphics Applications, Adam Levinthal, Pat Hanrahan, Mike Paquette, Jim Lawson, Pixar San Rafael, California, 1987.

Organization of the Motorola 88110 Superscalar RISC Microprocessor, Keith Diefendorff and Michael Allen.

**Microprocessor Report, vol. 7 No. 13, Oct. 4, 1993, IBM Regains Performance Lead with Power2, Six Way Superscalar CPU in MCM Achieves 126 SPECint92.**

(List continued on next page.)

(21) Appl. No.: 09/922,319

*Primary Examiner*—Matthew C. Bella

(22) Filed: **Aug. 2, 2001**

*Assistant Examiner—Mackly Monestime*

(65) **Prior Publication Data**

(74) *Attorney, Agent, or Firm*—McDermott, Will & Emery

US 2002/0133682 A1 Sep. 19, 2002

(57) **ABSTRACT**

### Related U.S. Application Data

(60) Continuation of application No. 09/382,402, filed on Aug. 24, 1999, now Pat. No. 6,295,599, which is a continuation-in-part of application No. 09/169,963, filed on Oct. 13, 1998, now Pat. No. 6,006,318, which is a continuation of application No. 08/754,827, filed on Nov. 22, 1996, now Pat. No. 5,822,603, which is a division of application No. 08/516,036, filed on Aug. 16, 1995, now Pat. No. 5,742,840.

The present invention provides a system and method for improving the performance of general purpose processors by expanding at least one source operand to a width greater than the width of either the general purpose register or the data path width. In addition, the present invention provides several classes of instructions which cannot be performed efficiently if the operands are limited to the width and accessible number of general purpose registers. The present invention provides operands which are substantially larger than the data path width of the processor by using a general purpose register to specify a memory address from which at least more than one, but typically several data path widths of data can be read. The present invention also provides for the efficient usage of a multiplier array that is fully used for high precision arithmetic, but is only partly used for other, lower precision operations.

(60) Provisional application No. 60/097,635, filed on Aug. 24, 1998.

(51) **Int. Cl.<sup>7</sup>** ..... **G06F 15/00**

(52) U.S. Cl. .... 712/210; 712/28; 712/24;  
712/32; 712/208

(58) Field of Search ..... 712/32, 28, 34,  
712/208, 210, 20, 24

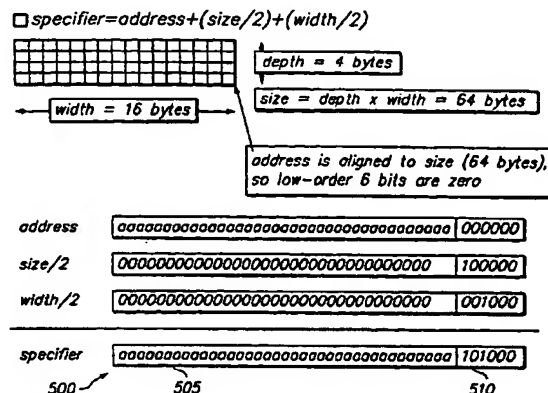
(56) **References Cited**

## U.S. PATENT DOCUMENTS

4,025,772	A	5/1977	Constant
4,489,393	A	12/1984	Kawahara et al.

**48 Claims, 148 Drawing Sheets**

**Microfiche Appendix Included  
(5 Microfiche, 63 Pages)**



## US 6,725,356 B2

Page 2

## U.S. PATENT DOCUMENTS

4,727,505 A 2/1988 Konishi et al.  
 4,876,660 A 10/1989 Owens et al.  
 4,893,267 A 1/1990 Alsup et al.  
 4,956,801 A 9/1990 Priem et al.  
 4,969,118 A 11/1990 Montoye et al.  
 4,975,868 A 12/1990 Freerksen  
 5,032,865 A 7/1991 Schlunt  
 5,157,388 A 10/1992 Kohn  
 5,201,056 A 4/1993 Daniel et al.  
 5,268,855 A 12/1993 Mason et al.  
 5,268,995 A 12/1993 Diefendorff et al.  
 5,280,598 A \* 1/1994 Osaki et al. .... 710/127  
 5,408,581 A 4/1995 Suzuki et al.  
 5,423,051 A 6/1995 Fuller et al.  
 5,426,600 A 6/1995 Nakagawa et al.  
 5,487,024 A \* 1/1996 Girardeau, Jr. .... 708/606  
 5,500,811 A 3/1996 Corry  
 5,557,724 A 9/1996 Sampat et al.  
 5,588,152 A 12/1996 Dapp et al.  
 5,592,405 A 1/1997 Gove et al.  
 5,600,814 A \* 2/1997 Gahan et al. .... 711/100  
 5,640,543 A 6/1997 Farrell et al.  
 5,642,306 A 6/1997 Mennemeier et al.  
 5,666,298 A 9/1997 Peleg et al.  
 5,669,010 A 9/1997 Duluk, Jr.  
 5,673,407 A 9/1997 Poland et al.  
 5,675,526 A 10/1997 Peleg et al.  
 5,680,338 A 10/1997 Agarwal et al.  
 5,721,892 A 2/1998 Peleg et al.  
 5,734,874 A 3/1998 Van Hook et al.  
 5,757,432 A 5/1998 Dulong et al.

5,758,176 A 5/1998 Agarwal et al.  
 5,768,546 A \* 6/1998 Kwon ..... 710/127  
 5,802,336 A 9/1998 Peleg et al.  
 5,809,292 A 9/1998 Wilkinson et al.  
 5,818,739 A 10/1998 Peleg et al.  
 5,825,677 A 10/1998 Agarwal et al.  
 5,835,782 A 11/1998 Lin et al.  
 5,886,732 A 3/1999 Humpleman  
 5,922,066 A 7/1999 Cho et al.  
 5,983,257 A 11/1999 Dulong et al.  
 6,016,538 A 1/2000 Guttag et al.  
 6,092,094 A 7/2000 Ireton  
 6,295,599 B1 \* 9/2001 Hansen et al. .... 712/32  
 6,401,194 B1 6/2002 Nguyen et al.

## OTHER PUBLICATIONS

IBM Creates PowerPC Processors for AS/400, Two New CPU's Implement 64-Bit Power PC with Extensions by Linley Gwennap, Jul. 31, 1995.

The Visual Instruction Set (VSI) in UltraSPAR™, L. Kohn, G. Maturana, M. Tremblay, A. Prabhu, G. Zyncer, May 3, 1995.

Osborne McGraw-Hill, i860™ Microprocessor Architecture, Neal Margulis, Foreword by Les Kohn.

A General-Purpose Array Processor for Seismic Processing, Nov.-Dec., 1984, vol. 1, No. 3) Revisiting past digital signal processor technology, Don Shaver- Jan.-Mar. 1998.

Accelerating Multimedia with Enhanced Microprocessors, Ruby B. Lee, 1995.

\* cited by examiner

U.S. Patent

Apr. 20, 2004

Sheet 1 of 148

US 6,725,356 B2

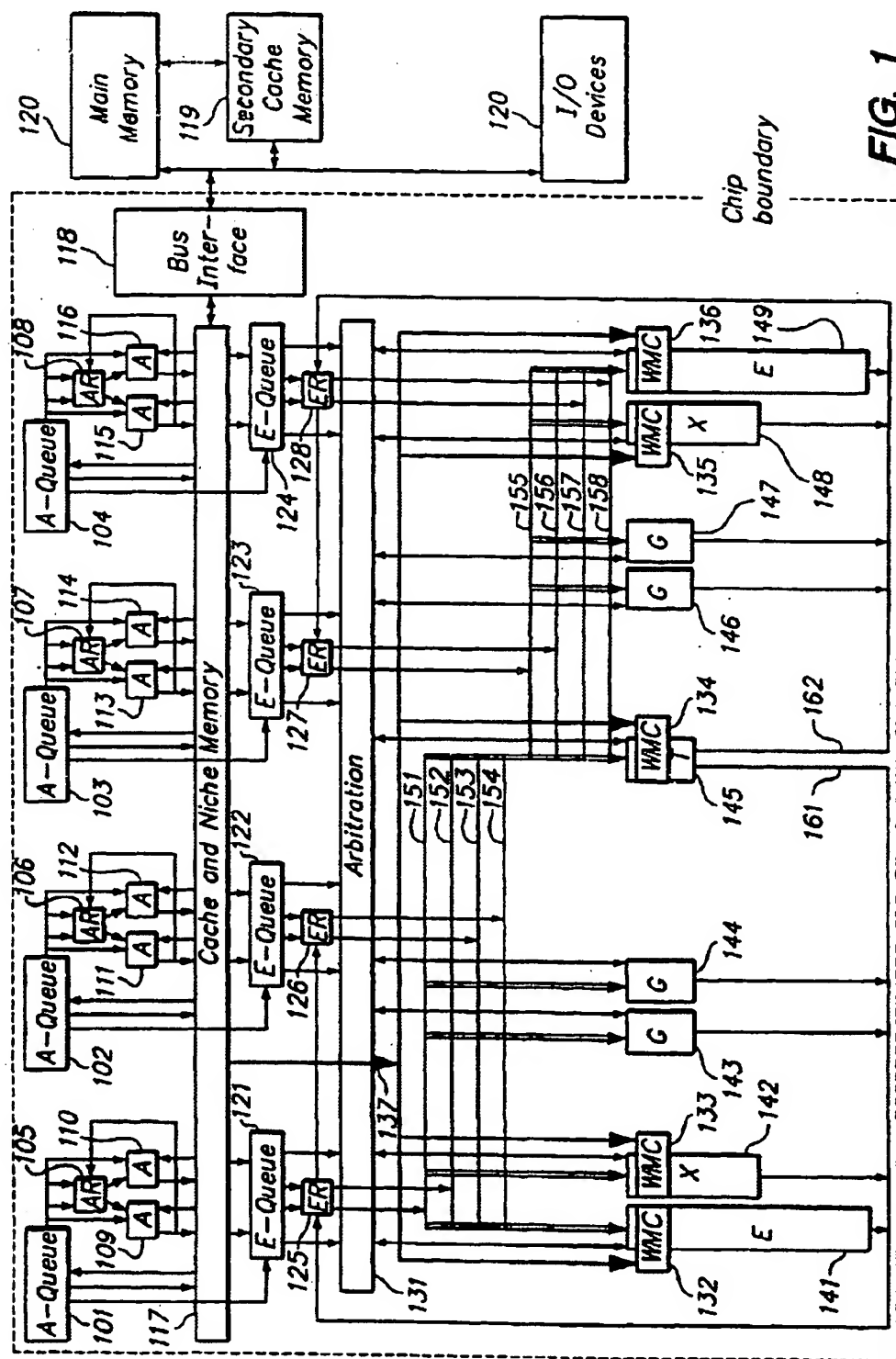


FIG. 1

U.S. Patent

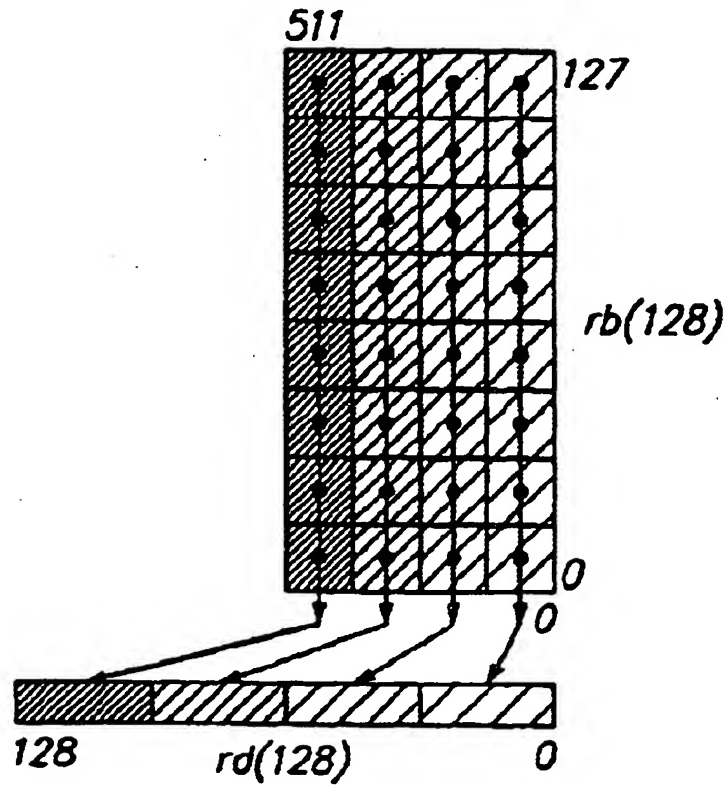
Apr. 20, 2004

Sheet 2 of 148

US 6,725,356 B2

$$\square rd_{128} = m[rc](128*64/size) * rb_{128}$$

$$m[rc](128*64/size)$$

**FIG. 2**

U.S. Patent

Apr. 20, 2004

Sheet 3 of 148

US 6,725,356 B2

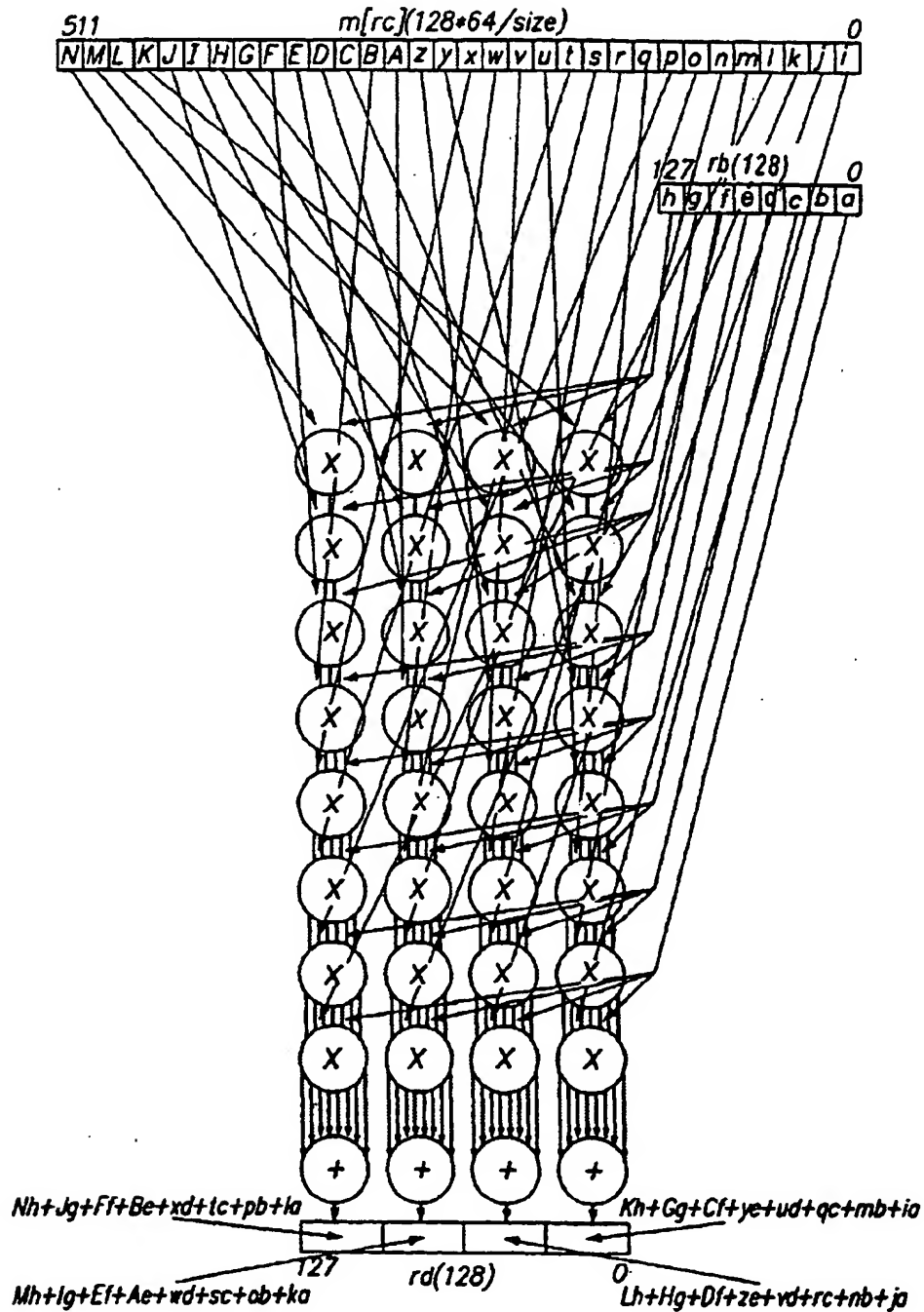


FIG. 3

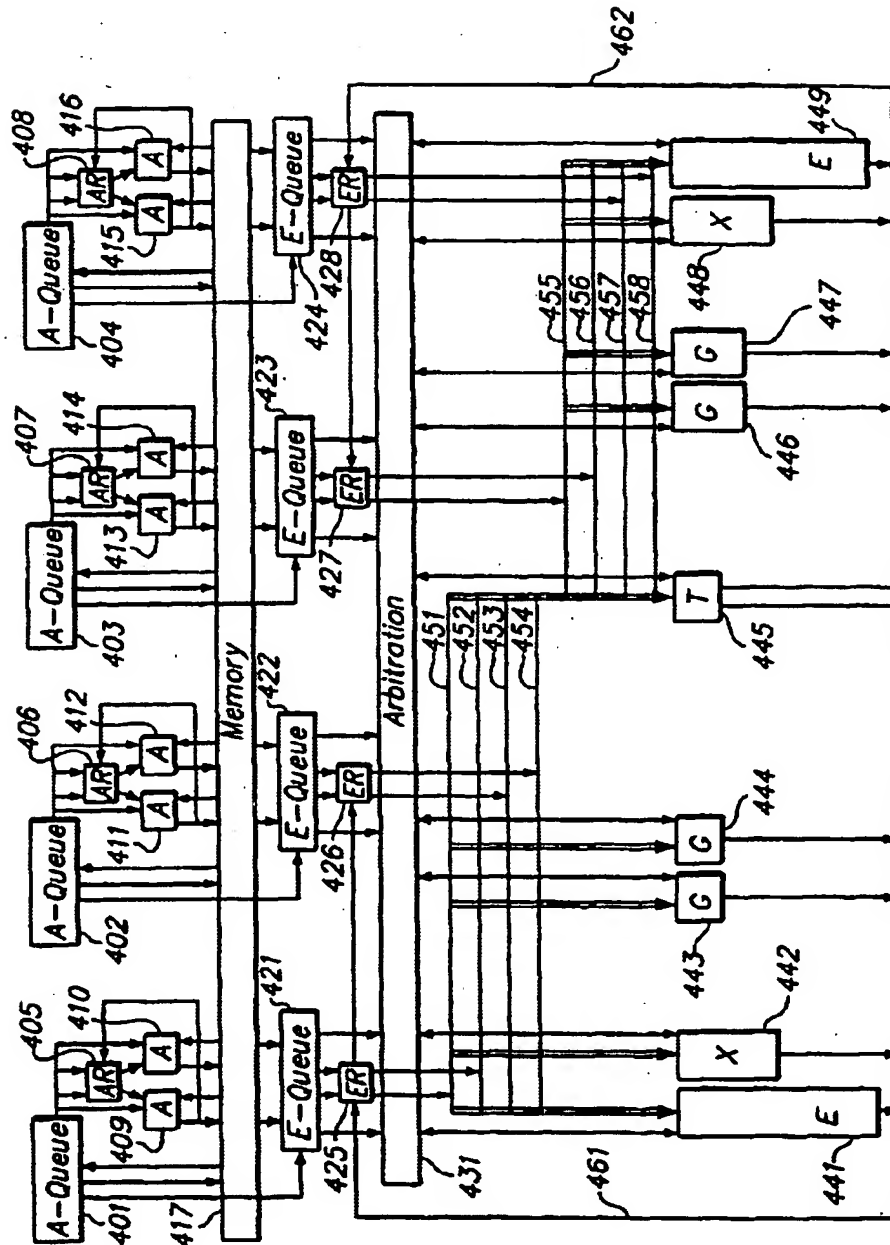
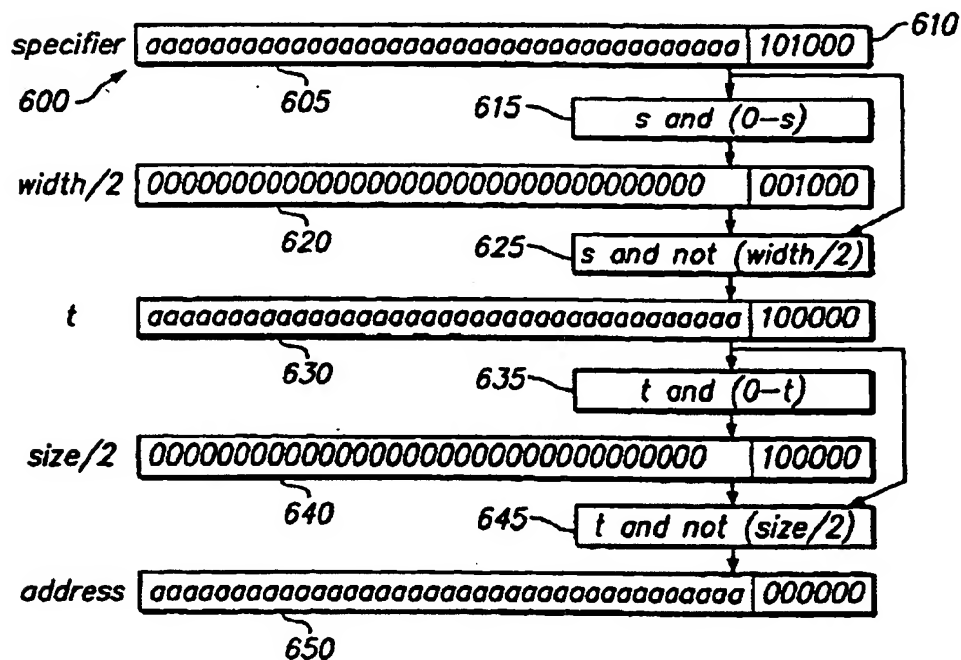
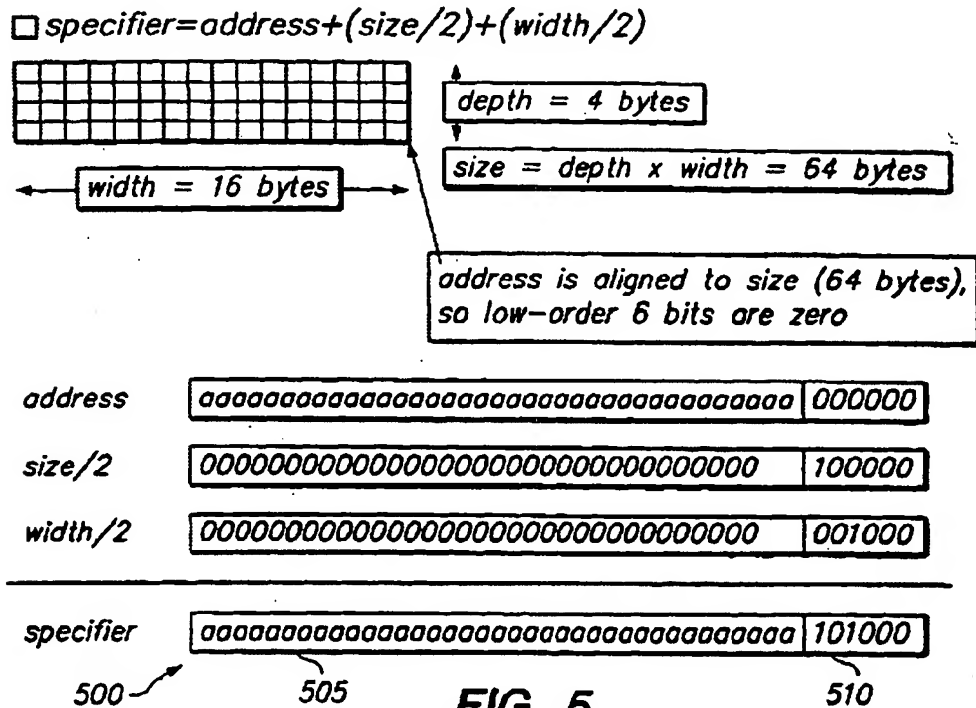


FIG. 4





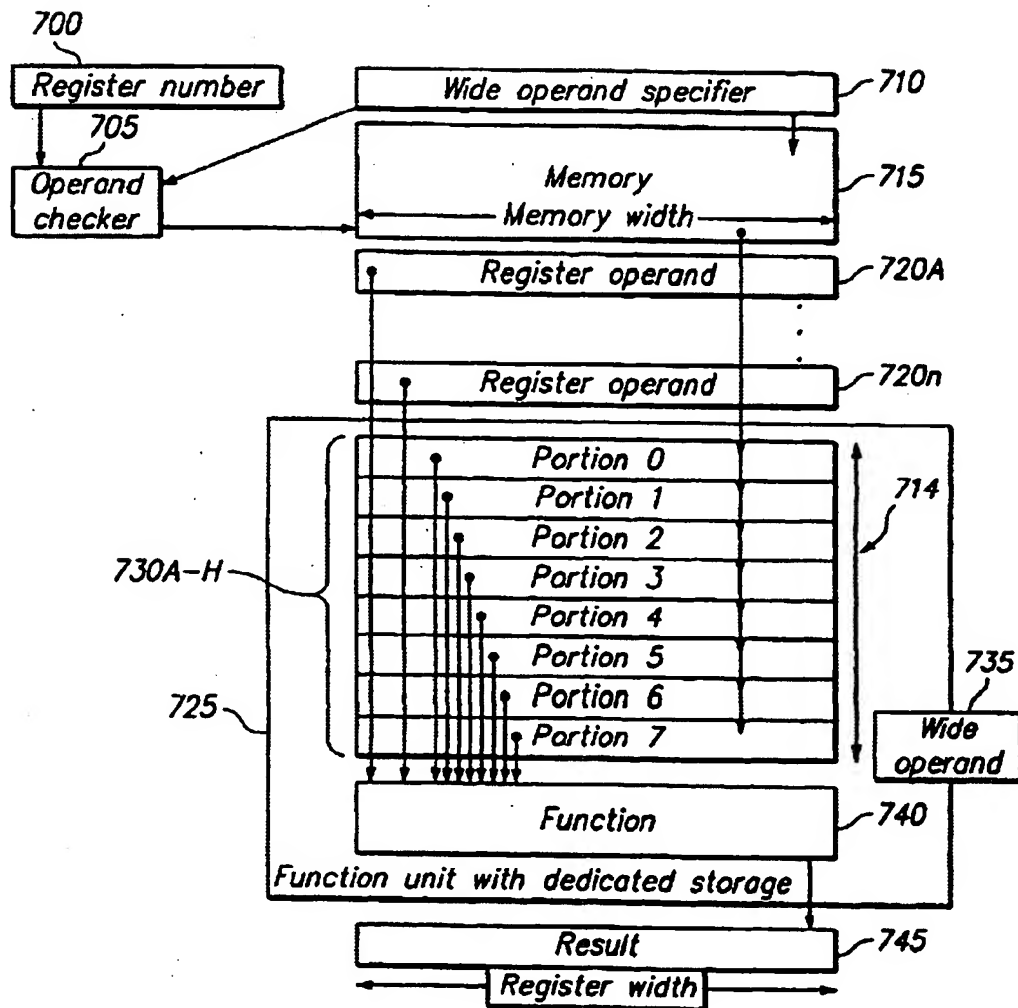
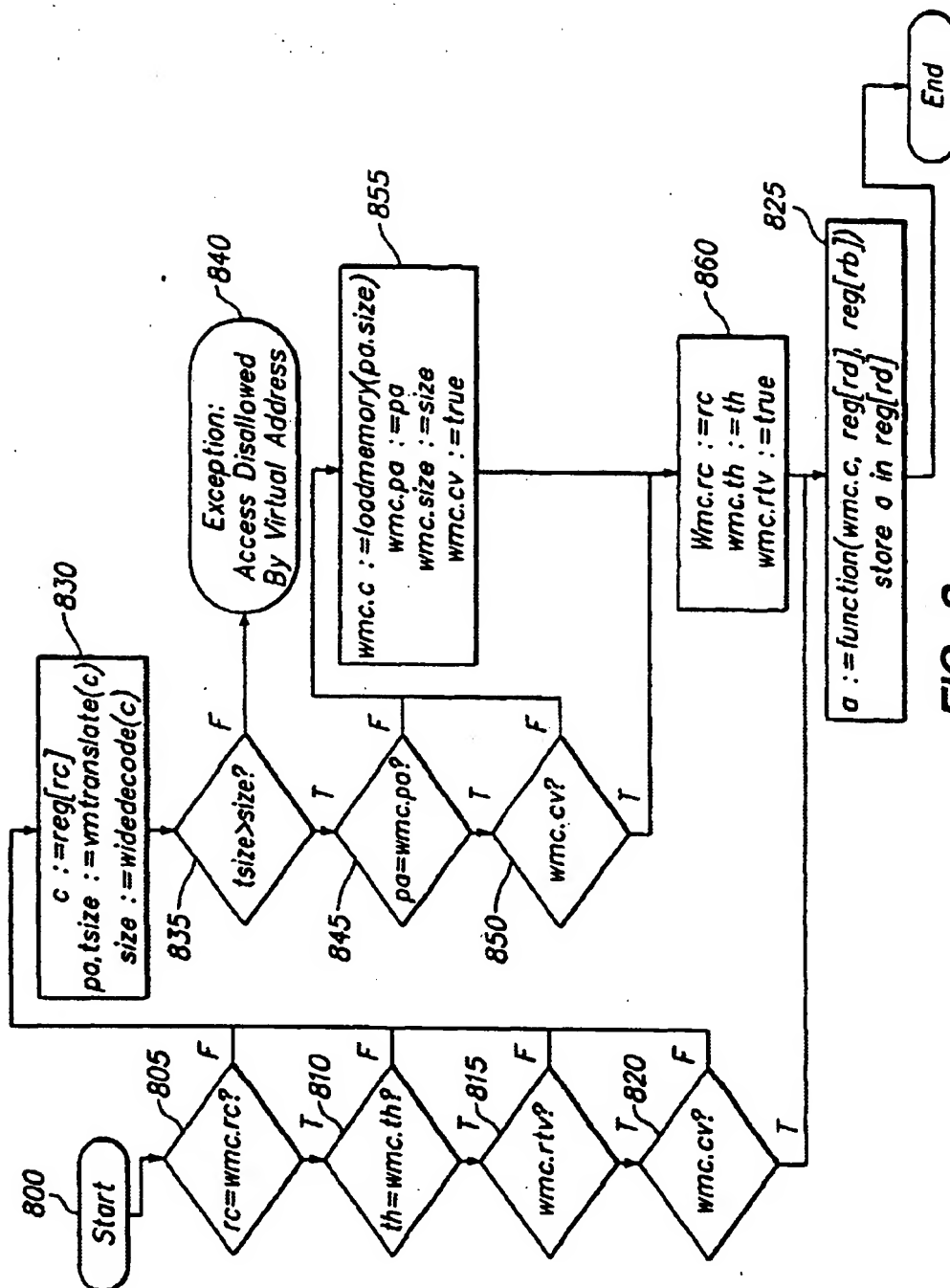


FIG. 7



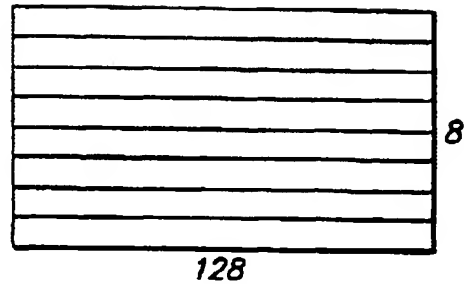
**U.S. Patent**

Apr. 20, 2004

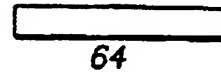
Sheet 8 of 148

**US 6,725,356 B2**

☐ *wmc.c* contents



☐ *wmc.pa*—physical address



☐ *wmc.size*—size of contents



☐ *wmc.cv*—contents valid



☐ *wmc.th*—thread last used



☐ *wmc.reg*—register last used



☐ *wmc.rtv*—register & thread valid



**FIG. 9**

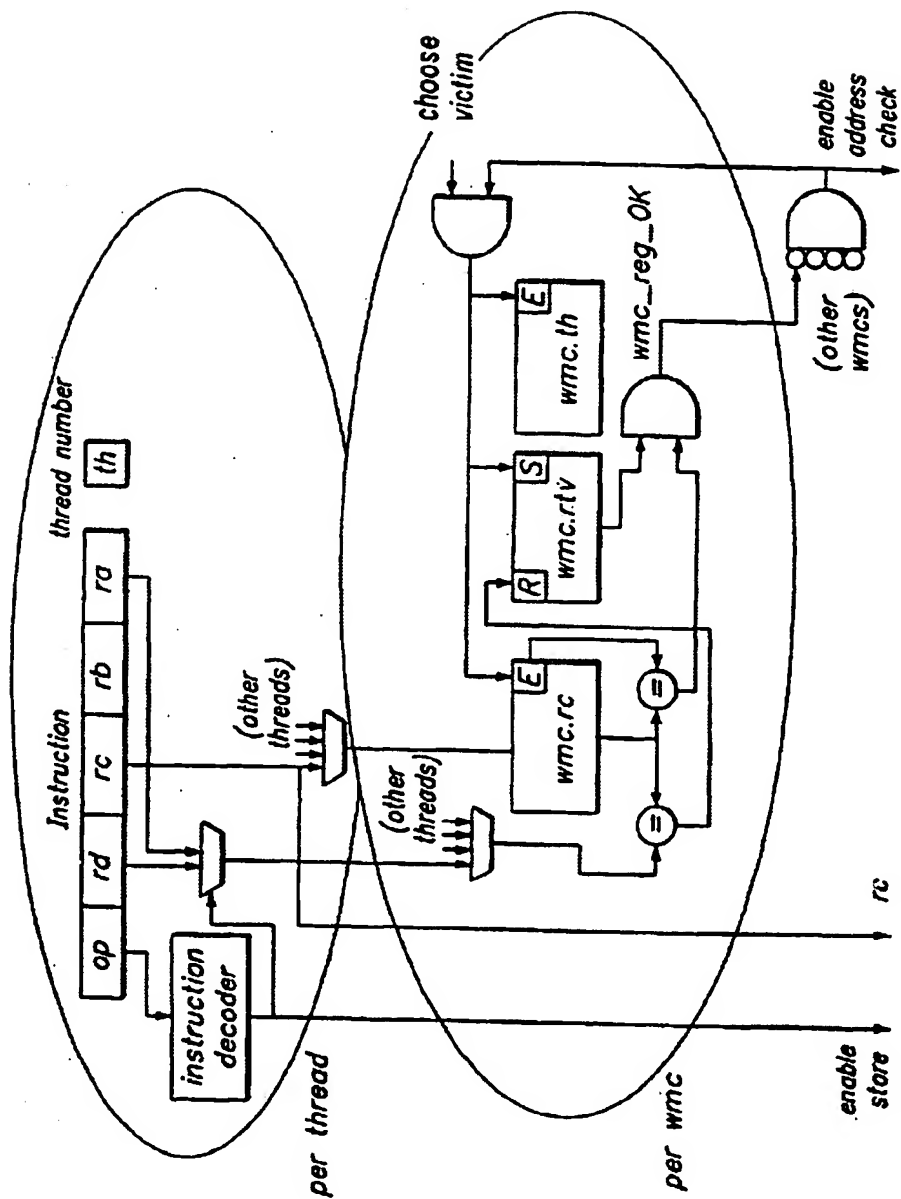


FIG. 10

U.S. Patent

Apr. 20, 2004

Sheet 10 of 148

US 6,725,356 B2

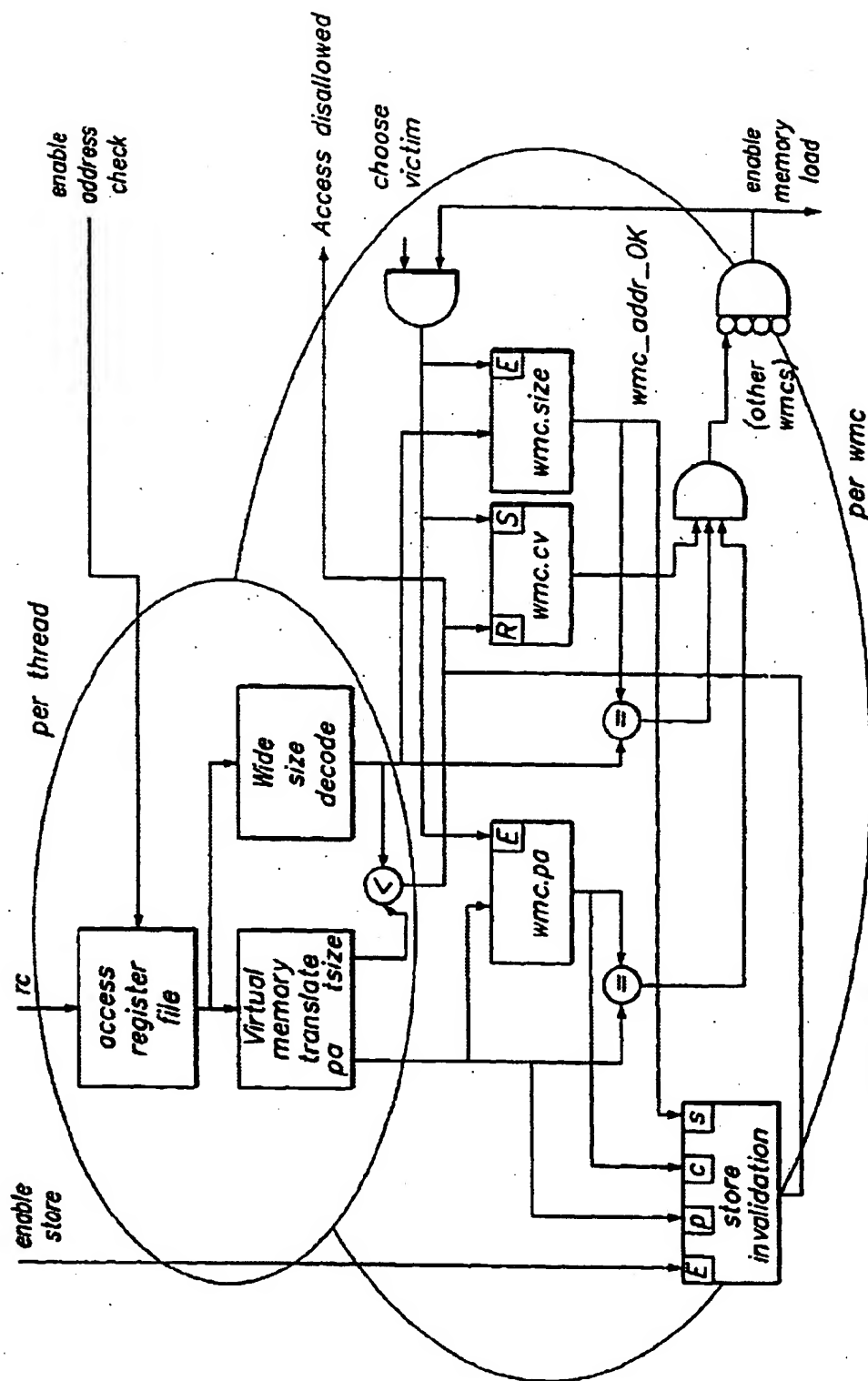


FIG. 11

**U.S. Patent**

Apr. 20, 2004

Sheet 11 of 148

**US 6,725,356 B2**

210

Operation codes

W.SWITCH.B	Wide switch big-endian
W.SWITCH.L	Wide switch little-endian

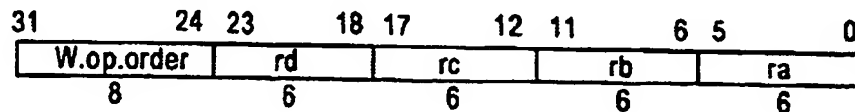
Selection

class	op	order
Wide switch	W.SWITCH	B L

Format

W.op.order ra=rc,rd,rb

ra=woporder(rc,rd,rb)

**FIG. 12A**

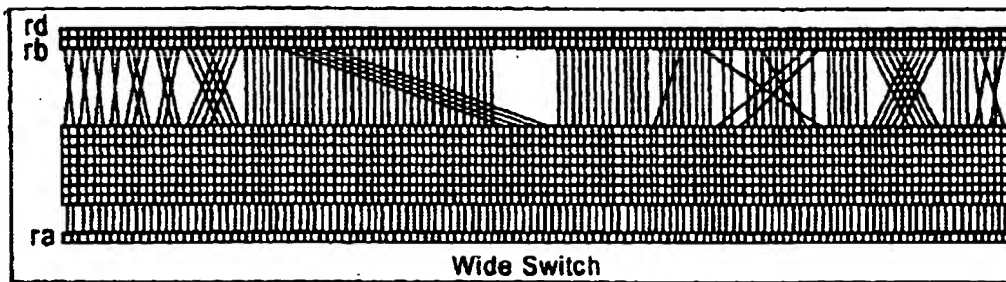
**U.S. Patent**

**Apr. 20, 2004**

**Sheet 12 of 148**

**US 6,725,356 B2**

1230



**FIG. 12B**

U.S. Patent

Apr. 20, 2004

Sheet 13 of 148

US 6,725,356 B2

1250

Definition

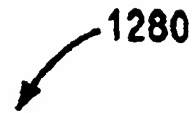
```

defWideSwitch(op,rd,rc,rb,ra)
  d ← RegRead(rd, 128)
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 128)
  if c1..0 ≠ 0 then
    raise AccessDisallowedByVirtual Address
  elseif c6..0 ≠ 0 then
    VirtAddr ← c and (c-1)
    W ← wsize ← (c and (0-c)) || 01
  else
    VirAddr ← c
    w ← wsize ← 128
  endif
  msize ← 8*wsize
  lwsiz ← log(wsize)
  case op of
    W.SWITCH.B:
      order ← B
    W.SWITCH.L:
      order ← L
  endcase
  m ← LoadMemory(c, VirtAddr,msize,order)
  db ← d || b
  for i ← 0 to 127
    j ← 0 || i1wsiz-1..0
    k ← m7*w+j || m6*w+j || m5*w+j || m4*w+j || m3*w+j || m2*w+j || mw+j || mj
    l ← i7..1wsiz || j1wsiz-1..0
    al ← dbj
  endfor
  RegWrite(ra, 128, a)
enddef

```

FIG. 12C



1280  


**Exceptions**

**Access disallowed by virtual address**  
**Access disallowed by tag**  
**Access disallowed by global TB**  
**Access disallowed by local TB**  
**Access detail required by tag**  
**Access detail required by local TB**  
**Access detail required by global TB**  
**Local TB miss**  
**Global TB miss**

**FIG. 12D**

**U.S. Patent**

Apr. 20, 2004

Sheet 15 of 148

**US 6,725,356 B2**

1210

Operation codes

W.TRANSLATE.8.B	Wide translate bytes big-endian
W.TRANSLATE.16.B	Wide translate doublets bit-endian
W.TRANSLATE.32.B	Wide translate quadlets bit-endian
W.TRANSLATE.64.B	Wide translate octlets big-endian
W.TRANSLATE.8.L	Wide translate bytes little-endian
W.TRANSLATE.16.L	Wide translate doublets little-endian
W.TRANSLATE.32.L	Wide translate quadlets little-endian
W.TRANSLATE.64.L	Wide translate octlets little-endian

Selection

class	size	order
Wide translate	8 16 32 64	B L

Format

W.TRANSLATE.size.order rd=rc,rb

rd=wtranslatesizeorder(rc,rb)

31	2434	1817	1211	65	21	0
W.TRANSLATE.order	rd	rc	rb	0	sz	
6	6	6	6	4	2	

sz ← log(size) = 3

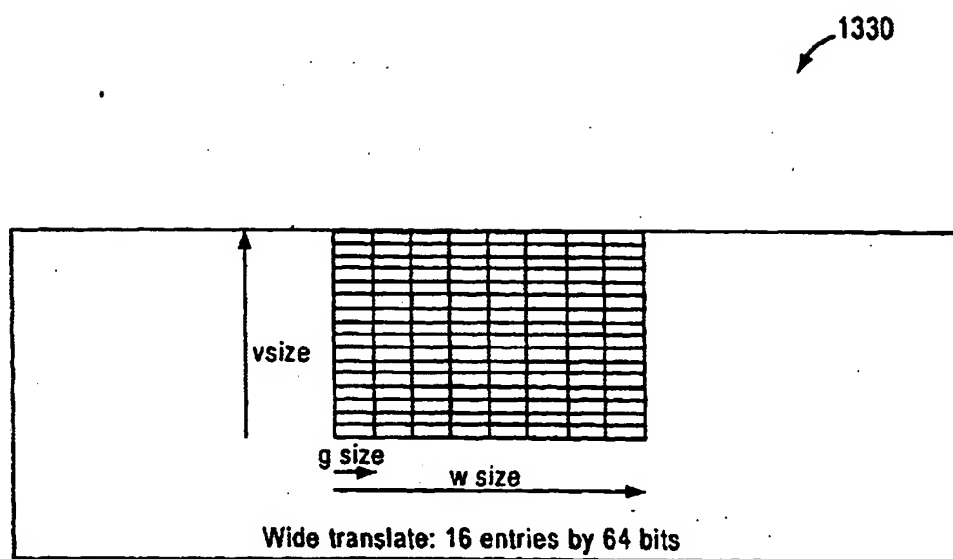
**FIG. 13A**

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 16 of 148**

**US 6,725,356 B2**



**FIG. 13B**

U.S. Patent

Apr. 20, 2004

Sheet 17 of 148

US 6,725,356 B2

1350

Definition

```

def Wide Translate(op, gsize, rd, rc, rb)
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 128)
  lsize ← log(gsize)
  if clsize-4..0 ≠ 0 then
    raise AccessDisallowedByVirtual Address
  endif
  if c4..lsize-3 ≠ 0 then
    wsize ← (c and (0-c)) || 03
    t ← c and (c-1)
  else
    wsize ← 128
    t ← c
  endif
  lsize ← log(wsize)
  if tlsize+4..lsize-2 ≠ 0 then
    msize ← (t and (0-t)) || 04
    VirtAddr ← t and (t-1)
  else
    msize ← 256*wsize
    VirtAddr ← t
  endif
  case op of
    W.TRANSLATE.B:
      order ← B
    W.TRANSLATE.L:
      order ← L
  endcase
  m ← LoadMemory(c, VirtAddr, msize, order)
  vsize ← msize/wsize
  lsize ← log(vsize)
  for i ← 0 to 128-gsize by gsize
    j ← ((order=B)lsize (blsize-1+1..i)) * wsize + ilsize-1..0
    agsize-1+1..i ← mj+gsize-1..j
  endfor
  RegWrite(rd, 128, a)
enddef

```

FIG. 13C

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 18 of 148**

**US 6,725,356 B2**

**1380**



**Exceptions**

**Access disallowed by virtual address**

**Access disallowed by tag**

**Access disallowed by global TB**

**Access disallowed by local TB**

**Access detail required by tag**

**Access detail required by local TB**

**Access detail required by global TB**

**Local TB miss**

**Global TB miss**

**FIG. 13D**

U.S. Patent

Apr. 20, 2004

Sheet 19 of 148

US 6,725,356 B2

Operation codes

1410

W.MUL.MAT.8.B	Wide multiply matrix signed byte big-endian
W.MUL.MAT.8.L	Wide multiply matrix signed byte little-endian
W.MUL.MAT.16.B	Wide multiply matrix signed doublet big-endian
W.MUL.MAT.16.L	Wide multiply matrix signed doublet little-endian
W.MUL.MAT.32.B	Wide multiply matrix signed quadlet big-endian
W.MUL.MAT.32.L	Wide multiply matrix signed quadlet little-endian
W.MUL.MAT.C.8.B	Wide multiply matrix signed complex byte big-endian
W.MUL.MAT.C.8.L	Wide multiply matrix signed complex byte little-endian
W.MUL.MAT.C.16.B	Wide multiply matrix signed complex doublet big-endian
W.MUL.MAT.C.16.L	Wide multiply matrix signed complex doublet little-endian
W.MUL.MAT.M.8.B	Wide multiply matrix mixed-signed byte big-endian
W.MUL.MAT.M.8.L	Wide multiply matrix mixed-signed byte little-endian
W.MUL.MAT.M.16.B	Wide multiply matrix mixed-signed doublet big-endian
W.MUL.MAT.M.16.L	Wide multiply matrix mixed-signed doublet little-endian
W.MUL.MAT.M.32.B	Wide multiply matrix mixed-signed quadlet big-endian
W.MUL.MAT.M.32.L	Wide multiply matrix mixed-signed quadlet little-endian
W.MUL.MAT.P.8.B	Wide multiply matrix polynomial byte big-endian
W.MUL.MAT.P.8.L	Wide multiply matrix polynomial byte little-endian
W.MUL.MAT.P.16.B	Wide multiply matrix polynomial doublet big-endian
W.MUL.MAT.P.16.L	Wide multiply matrix polynomial doublet little-endian
W.MUL.MAT.P.32.B	Wide multiply matrix polynomial quadlet big-endian
W.MUL.MAT.P.32.L	Wide multiply matrix polynomial quadlet little-endian
W.MUL.MAT.U.8.B	Wide multiply matrix unsigned byte big-endian
W.MUL.MAT.U.8.L	Wide multiply matrix unsigned byte little-endian
W.MUL.MAT.U.16.B	Wide multiply matrix unsigned doublet big-endian
W.MUL.MAT.U.16.L	Wide multiply matrix unsigned doublet little-endian
W.MUL.MAT.U.32.B	Wide multiply matrix unsigned quadlet big-endian
W.MUL.MAT.U.32.L	Wide multiply matrix unsigned quadlet little-endian

Selection

class	op	type	size	order
multiply	W.MUL.MAT	NONE MUP	8 16 32	B
				L
		C	8 16	B L

Format

W.op.size.order rd=rc,rb

rd=wopsizeorder(rc,rb)

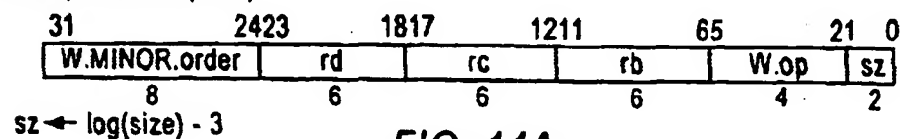


FIG. 14A

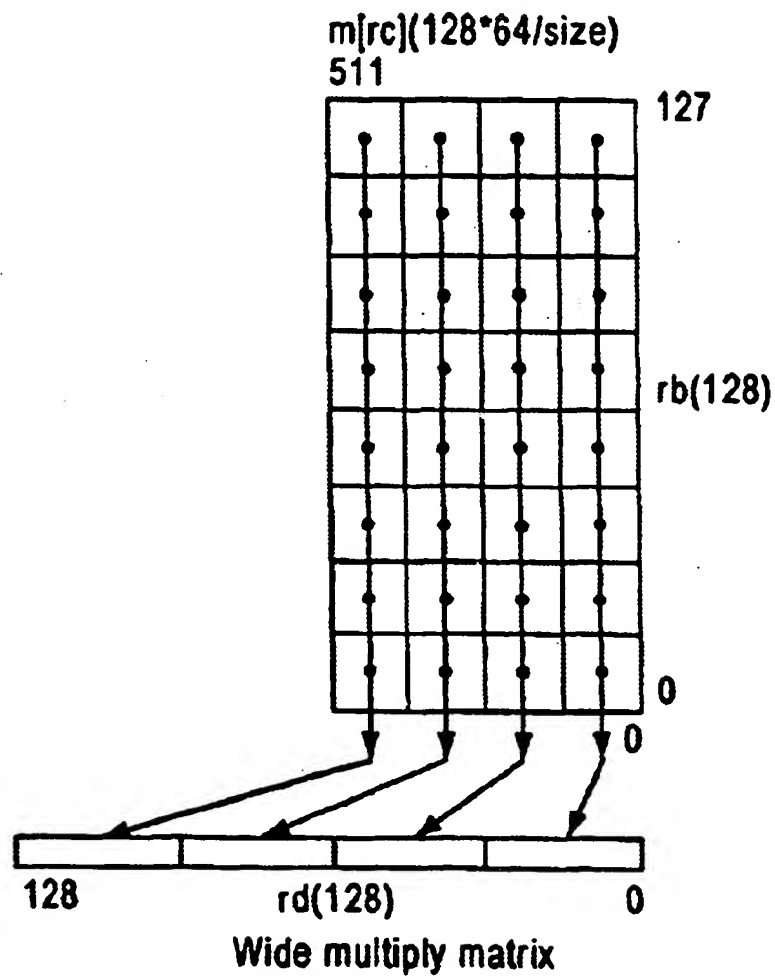
**U.S. Patent**

Apr. 20, 2004

Sheet 20 of 148

**US 6,725,356 B2**

1430



**FIG. 14B**

U.S. Patent

Apr. 20, 2004

Sheet 21 of 148

US 6,725,356 B2

1460

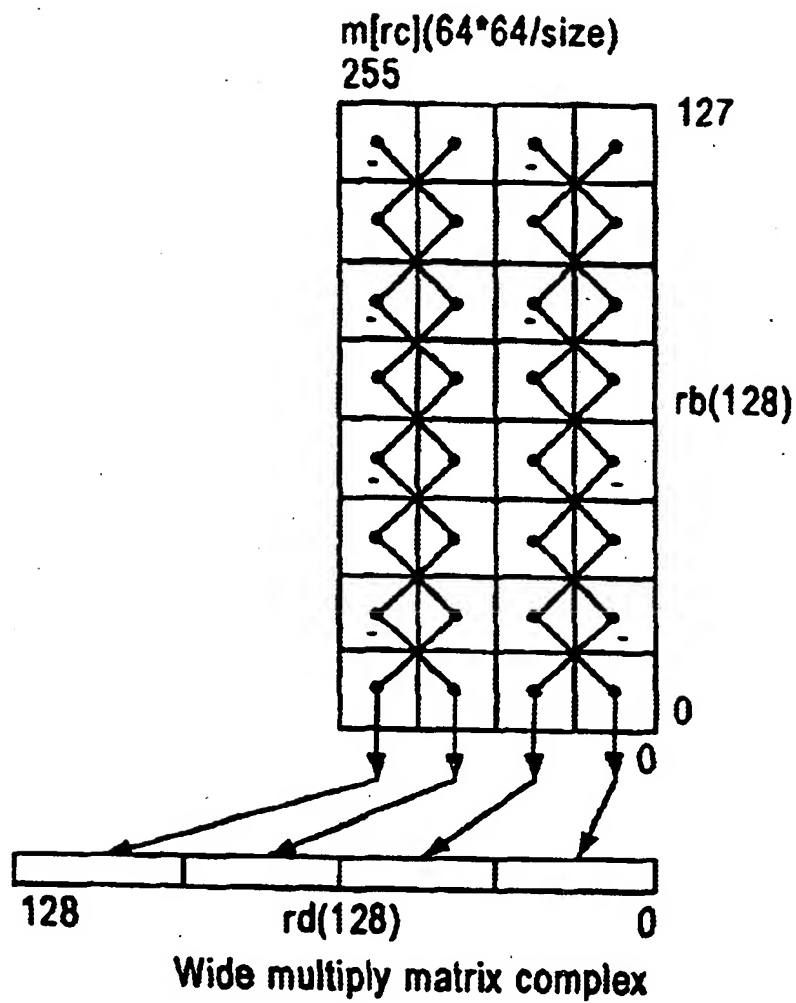


FIG. 14C



Definition

1480

```

def mul(size,h,vs,v,i,ws,j) as
  mul ← ((vs&vsize-1+i)h-size || vsize-1+i...i) * ((ws&wsiz-1+j)h-size || wsiz-1+j...j)
enddef

```

```

def c ← PolyMultiply(size,a,b) as
  p[0] ← 02*size
  for k ← 0 to size-1
    p[k+1] ← p[k] ^ ak ? (0size-k || b || 0k) : 02*size
  endfor
  c ← p[size]
enddef

```

```

def WideMultiplyMatrix(major,op,gsiz,rd,rc,rb)
  d ← RegRead(rd, 128)
  c ← RegRead(rc, 64)
  b ← RegRead(rb,128)
  lgsiz ← log(gsiz)
  if Clgsiz-4..0 ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  if C2..lgsiz-3 ≠ 0 then
    wsize ← (c and (0-c)) || 04
    t ← c and (c-1)
  else
    wsize ← 64
    t ← a
  endif
  lwsiz ← log(wsize)
  if tlwsiz+6-lgsiz..lwsiz-3 ≠ 0 then
    msiz ← (t and (0-t)) || 04
    VirtAddr ← t and (t-1)
  else
    msiz ← 128*wsize/gsiz
    VirtAddr ← t
  endif
  case major of
    W.MINOR.B:
      order ← B
    W.MINOR.L:
      order ← L
  endcase

```

FIG. 14D-1

U.S. Patent

Apr. 20, 2004

Sheet 23 of 148

US 6,725,356 B2

1480

```

case op of
  M.MUL.MAT.U.8, W.MUL.MAT.U.16, W.MUL.MAT.U.32,
  W.MUL.MAT.U.64:
    ms ← bs ← 0
  W.MUL.MAT.M.8, W.MUL.MAT.M.16, W.MUL.MAT.M.32,
  W.MUL.MAT.M.64:
    ms ← 0
    bs ← 1
  W.MUL.MAT.8, W.MUL.MAT.16, W.MUL.MAT.32,
  W.MUL.MAT.64, W.MUL.MAT.C.8, W.MUL.MAT.C.16,
  W.MUL.MAT.C.32, W.MUL.MAT.C.64:
    ms ← bs ← 1
  W.MUL.MAT.P.8, W.MUL.MAT.P.16, W.MUL.MAT.P.32,
  W.MUL.MAT.P.64:
endcase
m ← LoadMemory(c,VirtAddr,msize,order)
h ← 2*gszsize

for i ← 0 to wsize-gsize by gsize
  q[0] ← 02*gszsize
  for j ← 0 to vsize-gsize by gsize
    case op of
      W.MUL.MAT.P.8, W.MUL.MAT.P.16,
      W.MUL.MAT.P.32, W.MUL.MAT.P.64:
        k ← i+wsize*j8..lgszsize
        q[j+gszsize] ← q[j] ^ PolyMultiply(gsize,mk+gszsize-1..k,
        bj+gszsize-1..j)
      W.MUL.MAT.C.8, W.MUL.MAT.C.16, W.MUL.MAT.C.32,
      W.MUL.MAT.C.64:
        if (~i) & gsize = 0 then
          k ← i-(j&gszsize)+wsize*j8..lgszsize+1
          q[j+gszsize] ← q[i] + mul(gsize,h,ms,m,k,bs,b,j)
        else
          k ← i+gszsize+wsize*j8..lgszsize+1
          q[i+gszsize] ← q[i] = mul(gsize,h,ms,m,k,bs,b,j)
        endif
  endfor
endfor

```

FIG. 14D-2

**U.S. Patent**

Apr. 20, 2004

Sheet 24 of 148

**US 6,725,356 B2**

1480

```

W.MUL.MAT.8, W.MUL.MAT.16, W.MUL.MAT.32,
W.MUL.MAT.64, W.MUL.MAT.M.8, W.MUL.MAT.M.16,
W.MUL.MAT.M.32, W.MUL.MAT.M.64, W.MUL.MAT.U.8,
W.MUL.MAT.U.16, W.MUL.MAT.U.32, W.MUL.MAT.U.64
    q[i+gsize] ← q[i] + mul(gsize,h,ms,m,i+wsiz*
    j8..lgsize,bs,b,i)
endfor
    a2*gsiz-1+2*i..2*i ← q[vsize]
endfor
a127..2*wsiz ← 0
RegWrite(rd, 128, a)
enddef

```

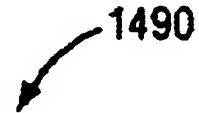
**FIG. 14D-3**

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 25 of 148**

**US 6,725,356 B2**

1490  


**Exceptions**

**Access disallowed by virtual address**

**Access disallowed by tag**

**Access disallowed by global TB**

**Access disallowed by local TB**

**Access detail required by tag**

**Access detail required by local TB**

**Access detail required by global TB**

**Local TB miss**

**Global TB miss**

**FIG. 14E**

**U.S. Patent**

Apr. 20, 2004

Sheet 26 of 148

**US 6,725,356 B2**

1510

Operation codes

W.MUL.MAT.X.B	Wide multiply matrix extract big-endian
W.MUL.MAT.X.L	Wide multiply matrix extract little-indian

Selection

class	op	order
Multiply matrix extract	W.MUL.MAT.X	B L

Format

W.op.order ra=rc,rd,rb

ra=wop(rc,rd,rb)

31	2423	1817	1211	65	0
W.op.order	rd	rc	rb	ra	
8	6	6	6	6	

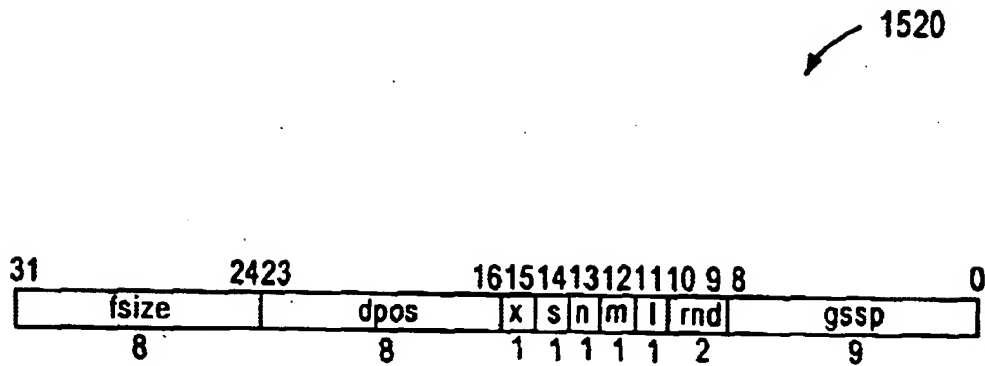
**FIG. 15A**

**U.S. Patent**

Apr. 20, 2004

Sheet 27 of 148

**US 6,725,356 B2**



**FIG. 15B**

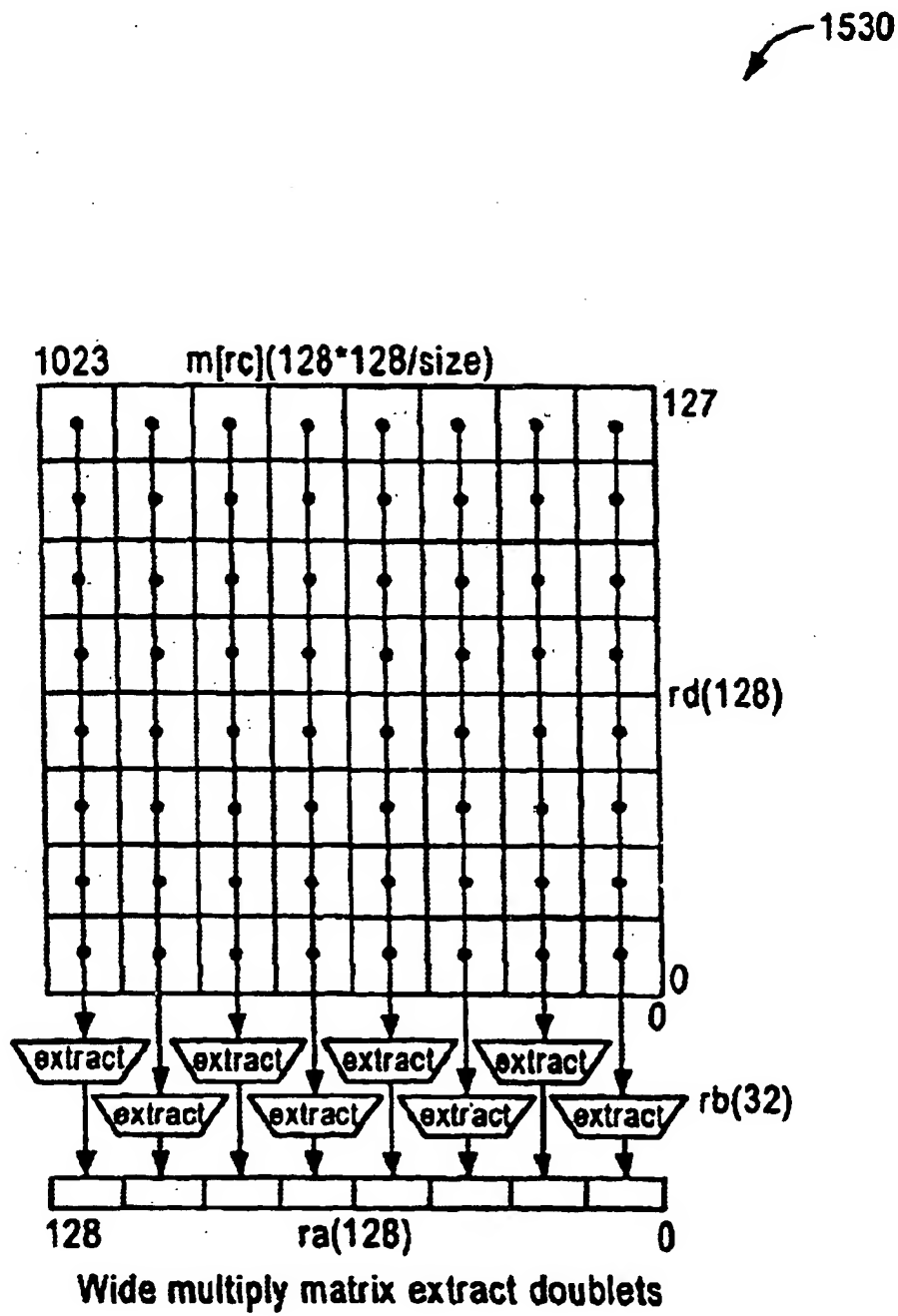


FIG. 15C

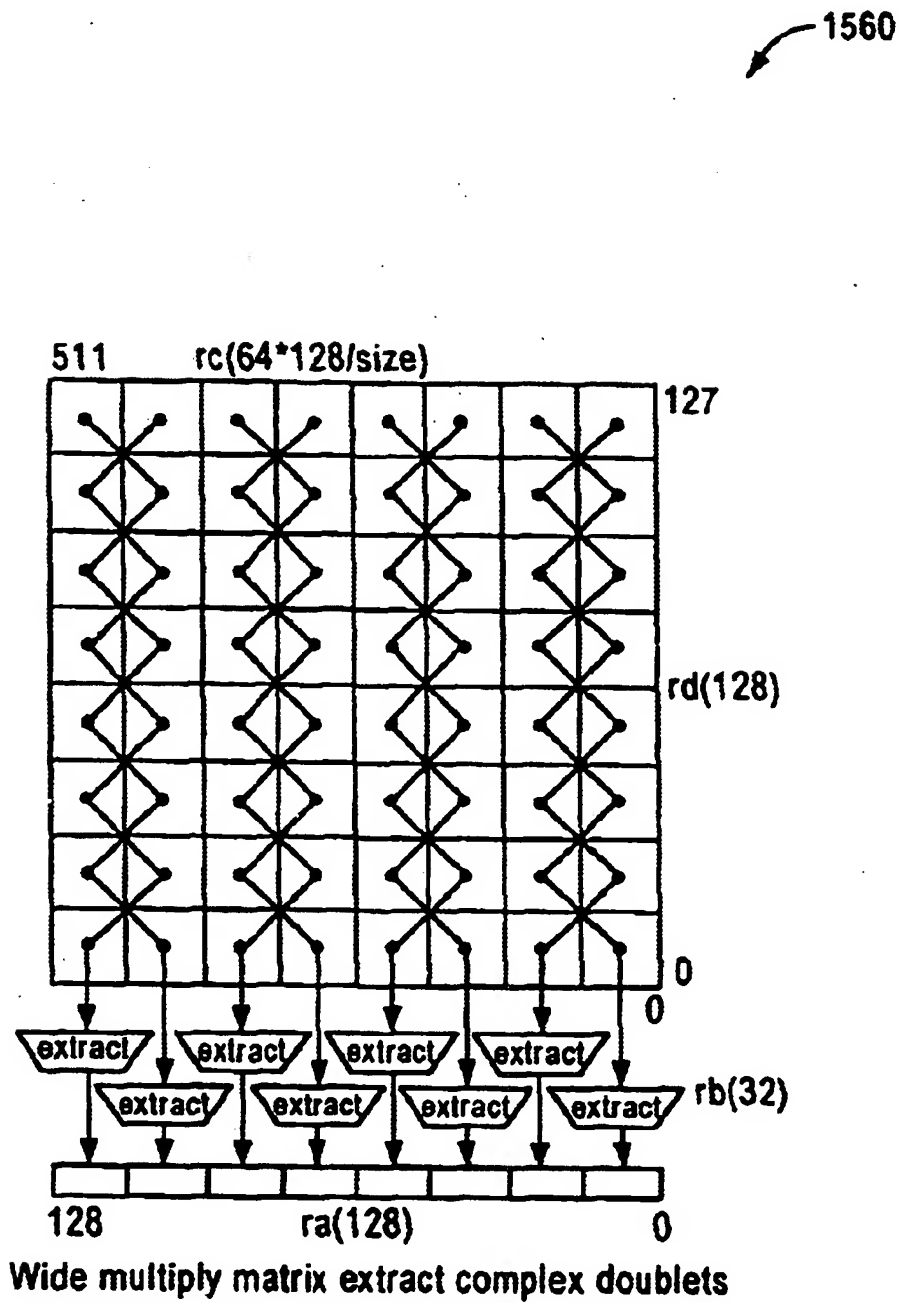


FIG. 15D



U.S. Patent

Apr. 20, 2004

Sheet 30 of 148

US 6,725,356 B2

Definition

```

def mul(size,h,vs,v,i,ws,w,j) as
    mul ← ((vs&vsize-1+i)h-size || vsize-1+i..i) * ((ws&wsize-1+j)h-size || wsize-1+j..j)
enddef

```

1580

```

def WideMultiplyMatrixExtract(op,ra,rb,rc,rd)

```

```

    d ← RegRead(rd, 128)
    c ← RegRead(rc, 64)
    b ← RegRead(rb, 128)
    case b8..0 of
        0..255:
            sgsz ← 128
        256..383:
            sgsz ← 64
        384..447:
            sgsz ← 32
        448..479:
            sgsz ← 16
        480..495:
            sgsz ← 8
        496..503:
            sgsz ← 4
        504..507:
            sgsz ← 2
        508..511:
            sgsz ← 1
    endcase
    l ← b11
    m ← b12
    n ← b13
    signed ← b14
    if c3..0 ≠ 0 then
        wsize ← (c and (0-c)) || 04
        t ← c and (c-1)
    else
        wsize ← 128
        t ← c
    endif
    if sgsz < 8 then
        gsize ← 8
    elseif sgsz > wsize/2 then
        gsize ← wsize/2
    else

```

FIG. 15E-1

U.S. Patent

Apr. 20, 2004

Sheet 31 of 148

US 6,725,356 B2

1580

```

    gsize ← sgsz
endif
lgsize ← log(gsize)
lwsz ← log(wsize)
if tlwsz+6-n-lgsize..lwsz-3 ≠ 0 then
    msz ← (t and (0-t)) || 04
    VirtAddr ← t and (t-1)
else
    msz ← 64*(2-n)*wsz/gsize
    VirtAddr ← t
endif
vsz ← (1+n)*msz*gsz/wsz
mm ← LoadMemory(c,VirtAddr,msz,order)
lmsz ← log(msz)
if (VirtAddrlmsz-4..0 ≠ 0) then
    raise AccessDisallowedByVirtualAddress
endif
case op of
    W.MUL.MAT.X.B:
        order ← B
    W.MUL.MAT.X.L:
        order ← L
endcase
ms ← signed
ds ← signed ^ m
as ← signed or m
spos ← (b8..0) and (2*gsz-1)
dpos ← (0 || b23..16) and (gsz-1)
r ← spos
sfsz ← (0 || b31..24) and (gsz-1)
tfsz ← (sfsz = 0) or ((sfsz+dpos) > gsz) ? gsz-dpos : sfsz
fsz ← (tfsz + spos > h) ? h - spos : tfsz
if (b10..9 = Z) & ~signed then
    rnd ← F
else
    rnd ← b10..9
endif

```

FIG. 15E-2

U.S. Patent

Apr. 20, 2004

Sheet 32 of 148

US 6,725,356 B2

1580

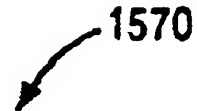
```

for i ← 0 to wsize-gsize by gsize
  q[0] ← 02*gsiz+7-lgsiz
  for j ← 0 to vsiz-gsiz by gsiz
    if n then
      if (~) & j & gsiz = 0 then
        k ← i-(j&gsiz)+wsiz*j8..lgsiz+1
        q[i+gsiz] ← q[i] + mul(gsiz,h,ms,mm,k,ds,d,j)
      else
        k ← i+gsiz+wsiz*j8..lgsiz+1
        q[i+gsiz] ← q[i] - mul(gsiz,h,ms,mm,k,ds,d,j)
      endif
    else
      q[i+gsiz] ← q[i] = mul(gsiz,h,ms,mm,i+j*wsiz/gsiz,ds,d,j)
    endif
  endfor
  p ← q[128]
  case rnd of
    none, N:
      s ← 0h-r || ~pr || prr-1
    Z:
      s ← 0h-r || ph-1r
    F:
      s ← 0h
    C:
      s ← 0h-r || 1r
  endcase
  v ← ((ds & ph-1) || p) + (0 || s)

  if (vh..r+fsiz = (as & vr+fsiz-1)h+1-r-fsiz) or not 1 then
    w ← (as & vr+fsiz-1)gsiz-fsiz-dpos || vfsiz-1+r..r || 0dpos
  else
    w ← (s ? (vh || ~vhgsiz-dpos-1) : 1gsiz-dpos) || 0dpos
  endif
  asize-1+i..i ← w
endfor
a127..wsiz ← 0
RegWrite(ra, 128, a)
enddef

```

FIG. 15E-3

1570  


**Exceptions**

**Access disallowed by virtual address**

**Access disallowed by tag**

**Access disallowed by global TB**

**Access disallowed by local TB**

**Access detail required by tag**

**Access detail required by local TB**

**Access detail required by global TB**

**Local TB miss**

**Global TB miss**

**FIG. 15F**

1610

Operation codes

W.MUL.MAT.X.I.8.B	Wide multiply matrix extract immediate signed byte big-endian
W.MUL.MAT.X.I.8.L	Wide multiply matrix extract immediate signed byte little-endian
W.MUL.MAT.X.I.16.B	Wide multiply matrix extract immediate signed doublet big-endian
W.MUL.MAT.X.I.16.L	Wide multiply matrix extract immediate signed doublet little-endian
W.MUL.MAT.X.I.32.B	Wide multiply matrix extract immediate signed quadlet big-endian
W.MUL.MAT.X.I.32.L	Wide multiply matrix extract immediate signed quadlet little-endian
W.MUL.MAT.X.I.64.B	Wide multiply matrix extract immediate signed octlets big-endian
W.MUL.MAT.X.I.64.L	Wide multiply matrix extract immediate signed octlets little-endian
W.MUL.MAT.X.I.C.8.B	Wide multiply matrix extract immediate complex bytes big-endian
W.MUL.MAT.X.I.C.8.L	Wide multiply matrix extract immediate complex bytes little-endian
W.MUL.MAT.X.I.C.16.B	Wide multiply matrix extract immediate complex doublets big-endian
W.MUL.MAT.X.I.C.16.L	Wide multiply matrix extract immediate complex doublets little-endian
W.MUL.MAT.X.I.C.32.B	Wide multiply matrix extract immediate complex quadlets big-endian
W.MUL.MAT.X.I.C.32.L	Wide multiply matrix extract immediate complex quadlets little-endian

Selection

class	op	type	size	order
wide multiply extract immediate	W.MUL.MAT.X.I	NONE	8 16 32 64	L B
		C	8 16 32	L B

Format

W.op.tsiz.order rd=rc,rb, i

rd=wopsizeorder(rc,rb,i)

31	24 23	18 17	12 11	6 5 4	32	0
W.op.order	rd	rc	rb	t	sz	sh
8	6	6	6	1	2	3

sz ← log(size) - 3

assert size+3 ≥ i ≥ size-4

sh ← i - size

FIG. 16A

1630

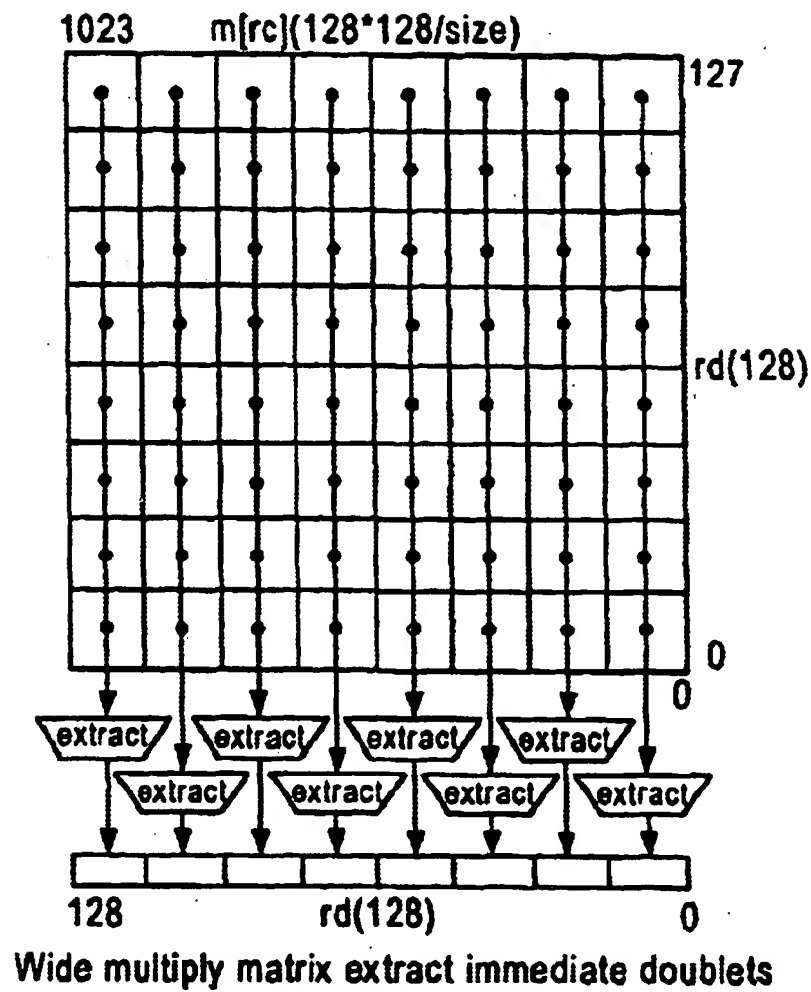
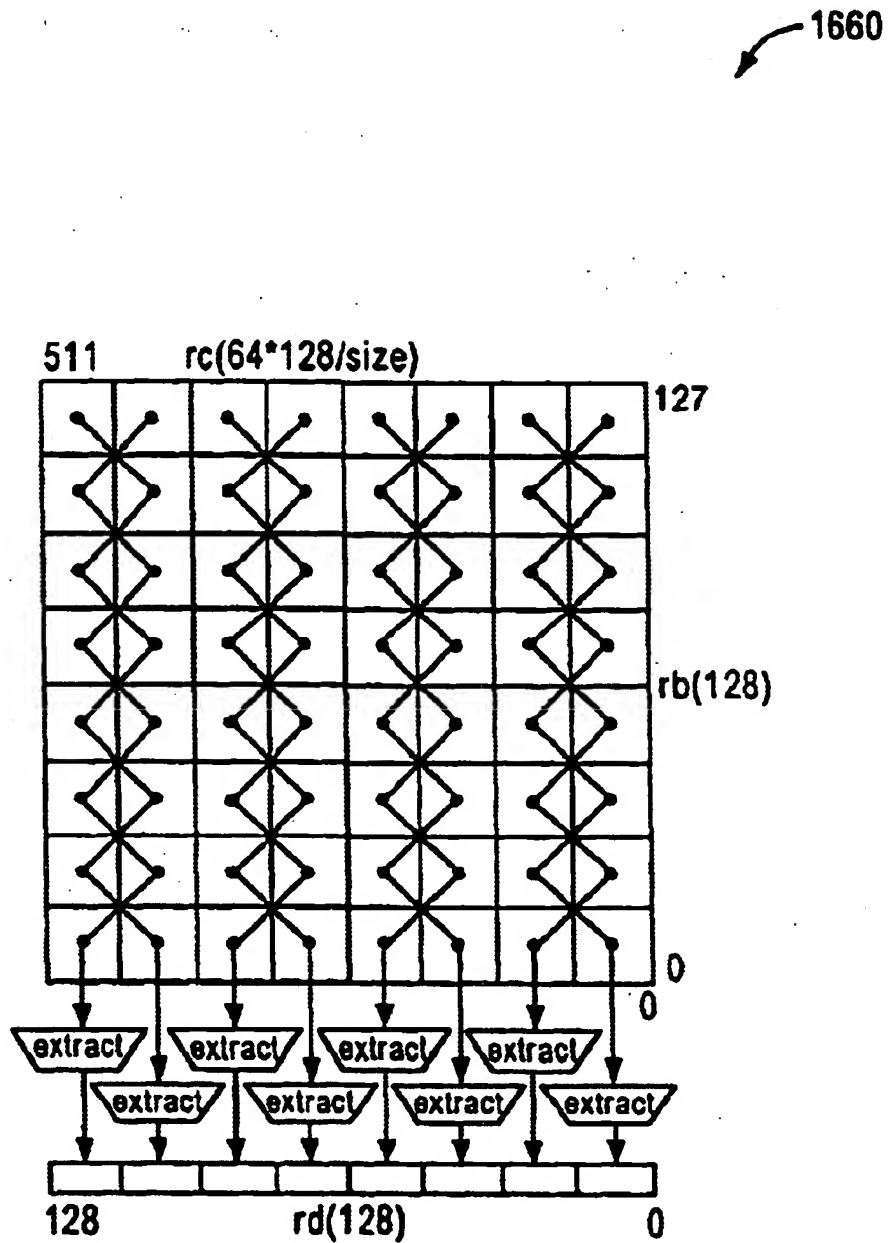


FIG. 16B



Wide multiply matrix extract immediate complex doublets

FIG. 16C

U.S. Patent

Apr. 20, 2004

Sheet 37 of 148

US 6,725,356 B2

1680

Definition

```
def mul(size,h,vs,v,i,ws,w,j) as
  mul ← ((vs&vsize-1+i)h-size || vsize-1+i..i) * ((ws&wsize-1+j)h-size || wsize-1+j..j)
enddef
```

```
def WideMultiplyMatrixExtractimmediate(op,type,gsize,rd,rc,rb,sh)
```

```
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 128)
  lsize ← log(gsize)
  case type of
    NONE:
      if clgsize-4..0 ≠ 0 then
        raise AccessDisallowedByVirtualAddress
      endif
      if c3..lgsize-3 ≠ 0 then
        wsize ← (c and (0-c)) || 04
        t ← c and (c-1)
      else
        wsize ← 128
        t ← c
      endif
      lwsiz ← log(wsize)
      if tlwsiz+6..lgsize..lwsiz-3 ≠ 0 then
        msize ← (t and (0-t)) || 04
        VirtAddr ← t and (t-1)
      else
        msize ← 128*wsize/gsize
        VirtAddr ← t
```

C:

```
  if clgsize-4..0 ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  if c3..lgsize-3 ≠ 0 then
    wsize ← (c and (0-c)) || 04
    t ← c and (c-1)
  else
    wsize ← 128
    t ← c
  endif
  lwsiz ← log(wsize)
  if tlwsiz+5..lgsize..lwsiz-3 ≠ 0 then
    msize ← (t and (0-t)) || 04
```

FIG. 16D-1



U.S. Patent

Apr. 20, 2004

Sheet 38 of 148

US 6,725,356 B2

```

VirtAddr ← t and (t-1)
else
  msize ← 64*wsz/gsz
  VirtAddr ← t
endif
vsz ← 2*msz*gsz/wsz
endcase
case of
  W.MUL.MAT.X.I.B:
    order ← B
  W.MUL.MAT.X.I.L:
    order ← L
endcase
as ← ms ← bs ← 1
m ← LoadMemory(c,VirtAddr,msz,order)
h ← (2*gsz) + 7 - lgsz - (ms and bs)
r ← gsz + (sh5||sh)
for ← 0 to wsz-gsz by gsz
  q[0] ← 02*gsz+7-lgsz
  for j ← 0 to vsz-gsz by gsz
    case type of
      NONE:
        q[j+gsz] ← q[j] + mul(gsz,h,ms,m,i+wsz*
          j8..lgsz,bs,b,j)
      C:
        if (~i) & j & gsz = 0 then
          k ← i-(j&gsz)+wsz*j8..lgsz+1
          q[j+gsz] ← q[j] + mul(gsz,h,ms,m,k,bs,b,j)
        else
          k ← i+gsz+wsz*j8..lgsz+1
          q[j+gsz] ← q[j] - mul(gsz,h,ms,m,k,bs,b,j)
        endif
      endcase
    endfor
    p ← q[vsz]
    s ← 0h-r||~pr|| pr-1
    v ← ((as & ph-1)||p) + (0||s)
    if (vh..r+gsz = (as & vr+gsz-1)h+1-r-gsz) then
      agsz-1+i..i ← vgsz-1+r..r
    else
      agsz-1+i..i ← as ? (vh||~vhgsz-1) : lgsz
    endif
  endfor
  a127..wsz ← 0
  RegWrite(rd, 128, a)
enddef

```

1680


FIG. 16D-2

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 39 of 148**

**US 6,725,356 B2**

1690  


**Exceptions**

**Access disallowed by virtual address**

**Access disallowed by tag**

**Access disallowed by global TB**

**Access disallowed by local TB**

**Access detail required by tag**

**Access detail required by local TB**

**Access detail required by global TB**

**Local TB miss**

**Global TB miss**

***FIG. 16E***

U.S. Patent

Apr. 20, 2004

Sheet 40 of 148

US 6,725,356 B2

1710

Operation codes

W.MUL.MAT.C.F.16.B	Wide multiply matrix complex floating-point half big-endian
W.MUL.MAT.C.F.16.L	Wide multiply matrix complex floating-point little-endian
W.MUL.MAT.C.F.32.B	Wide multiply matrix complex floating-point single big-endian
W.MUL.MAT.C.F.32.L	Wide multiply matrix complex floating-point single little-endian
W.MUL.MAT.F.16.B	Wide multiply matrix floating-point half big-endian
W.MUL.MAT.F.16.L	Wide multiply matrix floating-point half little-endian
W.MUL.MAT.F.32.B	Wide multiply matrix floating-point single big-endian
W.MUL.MAT.F.32.L	Wide multiply matrix floating-point single little-endian
W.MUL.MAT.F.64.B	Wide multiply matrix floating-point double big-endian
W.MUL.MAT.F.64.L	Wide multiply matrix floating-point double little-endian

Selection

class	op	type	prec	order
wide multiply matrix	W.MUL.MAT	F	16 32 64	L B
		C.F	16 32	L B

Format

W.op.prec.order rd=rc,rb

rd=woppreorder(rc,rb)

31	24 23	18 17	12 11	6 5	21	0
W.MINOR.order	rd	rc	rb	W.op	pr	
8	6	6	6	4	2	

Pr ← log(prec) - 3

FIG. 17A

U.S. Patent

Apr. 20, 2004

Sheet 41 of 148

US 6,725,356 B2

1730

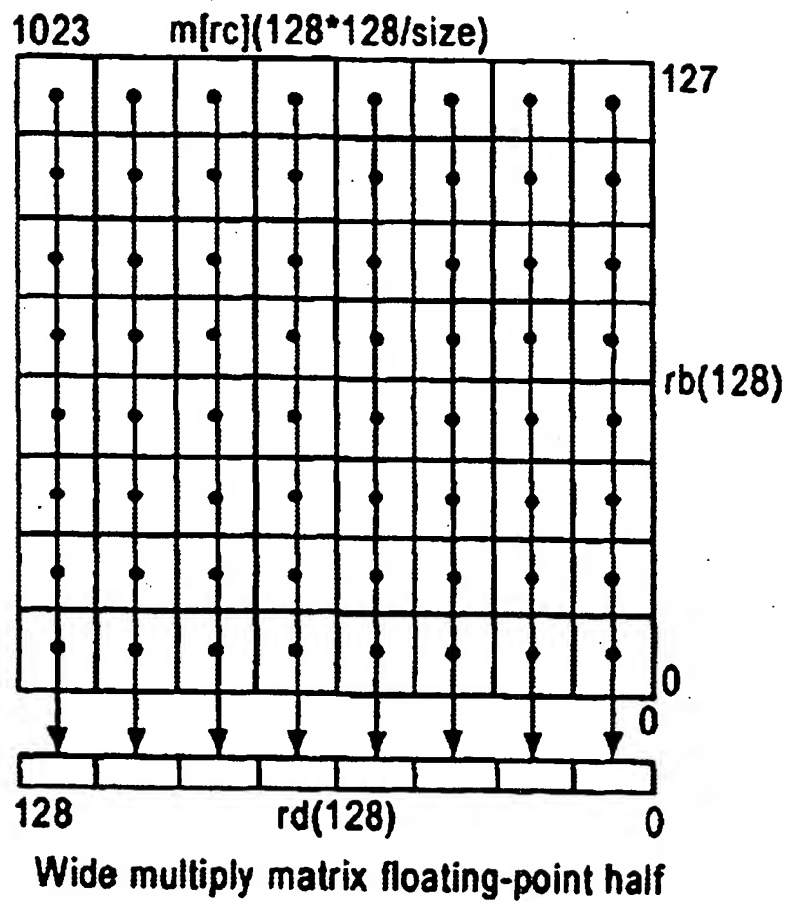
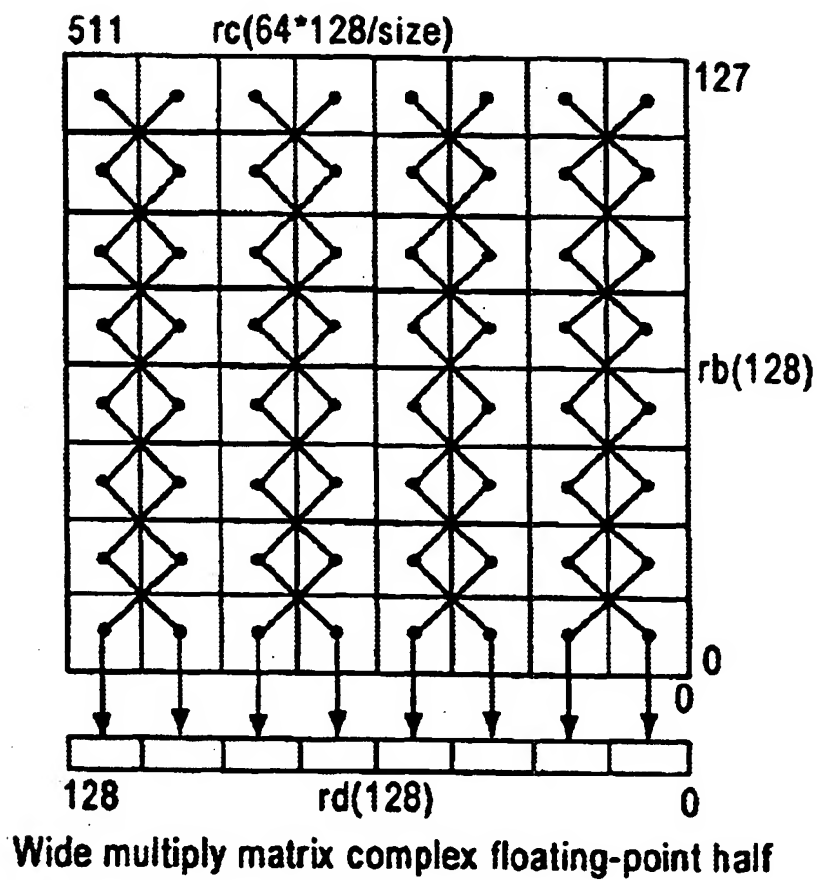


FIG. 17B

1760



**FIG. 17C**

U.S. Patent

Apr. 20, 2004

Sheet 43 of 148

US 6,725,356 B2

1780

Definition

```

def mul(size,v,i,w,j) as
    mul ← fmul(F(size,vsize-1+i..i),F(size,wsize-1+j..j))
enddef

def WideMultiplyMatrixFloatingPoint(major,op,gsize,rd,rc,rb)
    c ← RegRead(rc, 64)
    b ← RegRead(rb, 128)
    lgsiz ← log(gsize)
    switch op of
        W.MUL.MAT.F.16, W.MUL.MAT.F.32, W.MUL.MAT.F.64:
            if clgsiz-4..0 ≠ 0 then
                raise AccessDisallowedByVirtualAddress
            endif
            if c3..lgsiz-3 ≠ 0 then
                wsize ← (c and (0-c)) || 04
                t ← c and (c-1)
            else
                wsize ← 128
                t ← c
            endif
            lwsiz ← log(wsize)
            if tlwsiz-6-lgsiz..lwsiz-3 ≠ 0 then
                msize ← (t and (0-t)) || 04
                VirtAddr ← t and (t-1)
            else
                msize ← 128*wsize/gsize
                VirtAddr ← t
            endif
            vsize ← msize*gsize/wsize
        W.MUL.MAT.C.F.16, W.MUL.MAT.C.F.32, W.MUL.MAT.C.F.64:
            if clgsiz-4..0 ≠ 0 then
                raise AccessDisallowedByVirtualAddress
            endif
            if c3..lgsiz-3 ≠ 0 then
                wsize ← (c and (0-c)) || 04
                t ← c and (c-1)
            else
                wsize ← 128
                t ← c
            endif
            lwsiz ← log(wsize)
            if tlwsiz-5-lgsiz..lwsiz-3 ≠ 0 then

```

FIG. 17D-1

U.S. Patent

Apr. 20, 2004

Sheet 44 of 148

US 6,725,356 B2

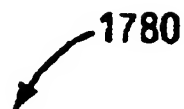
1780

```

        msize ← (t and (0-t)) || 04
        VirtAddr ← t and (t-1)
    else
        msize ← 64*wsz/gsize
        VirtAddr ← t
    endif
    vsz ← 2*msz*gsz/wsz
endcase
case major of
    M.MINOR.B:
        order ← B
    M.MINOR.L:
        order ← L
endcase
m ← LoadMemory(c,VirtAddr,msz,order)
for i ← 0 to wsz-gsz by gsz
    q[0].t ← NULL
    for j ← 0 to vsz-gsz by gsz
        case op of
            W.MUL.MAT.F.16, W.MUL.MAT.F.32, W.MUL.MAT.F.64:
                q[j+gsz] ← faddq[j], mul(gsz,m,i+wsz*
                    j8..lgsz+1,b,j))
            W.MUL.MAT.C.F.16, W.MUL.MAT.C.F.32,
            W.MUL.MAT.C.F.64:
                if (~i) & j & gsz = 0 then
                    k ← i-(j&gsz)+wsz*j8..lgsz+1
                    q[j+gsz] ← faqq[j], mul(gsz,m,k,b,j))
                else
                    k ← i+gsz+wsz*j8..lgsz+1
                    q[j+gsz] ← fsubq[j], mul(gsz,m,k,b,j))
                endif
            endif
        endcase
    endfor
    a[gsz-1+i..i] ← q[vsz]
endfor
a127..wsz ← 0
RegWrite(rd, 128, a)
enddef

```

FIG. 17D-2

**1780**  


**Exceptions**

**Floating-point arithmetic**  
**Access disallowed by virtual address**  
**Access disallowed by tag**  
**Access disallowed by global TB**  
**Access disallowed by local TB**  
**Access detail required by tag**  
**Access detail required by local TB**  
**Access detail required by global TB**  
**Local TB miss**  
**Global TB miss**


**FIG. 17E**



**U.S. Patent**

Apr. 20, 2004

Sheet 46 of 148

**US 6,725,356 B2**
 1810
Operation codes

W.MUL.MAT.G.8.B	Wide multiply matrix Galois bytes big-endian
W.MUL.MAT.G.8.L	Wide multiply matrix Galois bytes little-endian

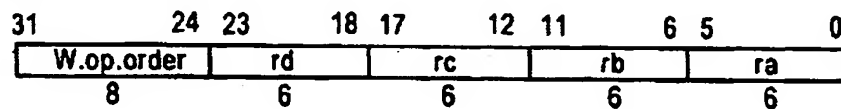
Selection

class	op	size	order
Multiply matrix Galois	W.MUL.MAT.G	8	B L

Format

W.op.order ra=rc,rd,rb

ra=woporder(rc,rd,rb)

**FIG. 18A**

U.S. Patent

Apr. 20, 2004

Sheet 47 of 148

US 6,725,356 B2

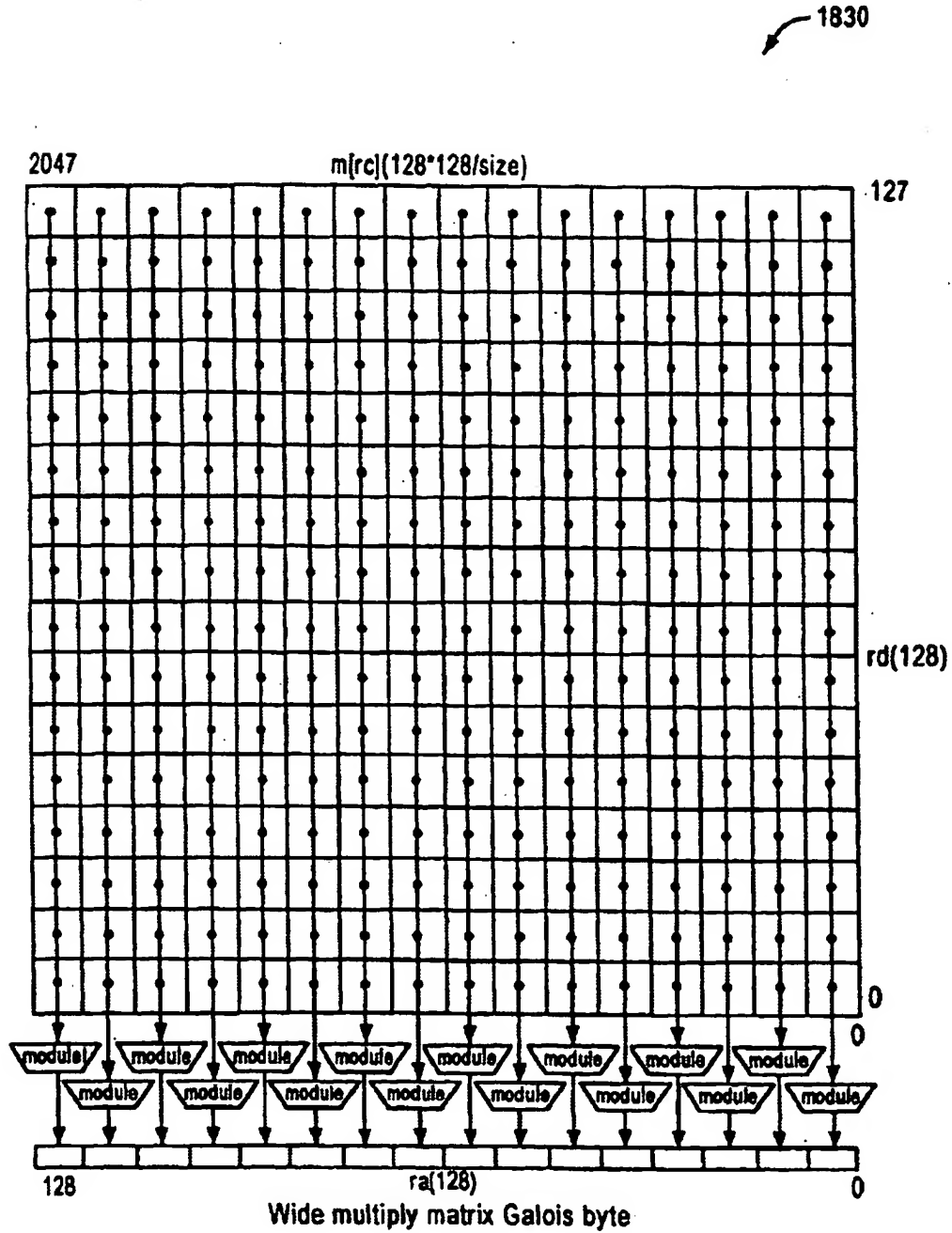


FIG. 18B

U.S. Patent

Apr. 20, 2004

Sheet 48 of 148

US 6,725,356 B2

Definition

1860

```

def c ← PolyMultiply(size,a,b) as
  p[0] ← 02*size
  for k ← 0 to size-1
    p[k+1] ← p[k] ^ ak ? (0size-k || b || 0k) : 02*size
  endfor
  c ← p[size]
enddef

def c ← PolyResidue(size,a,b) as
  p[0] ← a
  for k ← size-1 to 0 by -1
    p[k-1] ← p[k] ^ p[0]size+k ? (0size-k || 11 || b || 0k) : 02*size
  endfor
  c ← p[size]size-1..0
enddef

def WideMultiplyMatrixGalois(op,gsize,rd,rc,rb,ra)
  d ← RegRead(rd, 128)
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 128)
  lgsize ← log(gsize)
  if clgsize-4..0 ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  if c3..lgsize-3 ≠ 0 then
    wsize ← (c and (0-c)) || 04
    t ← c and (c-1)
  else
    wsize ← 128
    t ← c
  endif
  lwsz ← log(wsize)
  if tlwsz+6-lgsize..lwsz-3 ≠ 0 then
    msize ← (t and (0-t)) || 04
    VirtAddr ← t and (t-1)
  else
    msize ← 128*wsize/gsize
    VirtAddr ← t
  endif
  case op of
    W.MUL.MAT.G.8.B:
      order ← B
    W.MUL.MAT.G.8.L:
      order ← L
  endcase

```

FIG. 18C-1

**U.S. Patent**

Apr. 20, 2004

Sheet 49 of 148

**US 6,725,356 B2**

1860

```

m ← LoadMemory(c, VirtAddr, msize, order)
for i ← 0 wsize-gsize by gsize
  q[0] ← 02*gsize
  for j ← 0 to vsize-gsize by gsize
    k ← i+wsize*8..lgsize
    q[j+gsize] ← q[j] ^ PolyMultiply(gsize, mk←gsize-1..k, dj←gsize-1..j)
  endfor

  agsize-1+i..i ← PolyResidue(gsize, q[vsize], bgsize-1..0)
endfor
a127..wsize ← 0
RegWrite(ra, 128, a)
enddef

```


**FIG. 18C-2**

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 50 of 148**

**US 6,725,356 B2**

1890  


**Exceptions**

**Access disallowed by virtual address**

**Access disallowed by tag**

**Access disallowed by global TB**

**Access disallowed by local TB**

**Access detail required by tag**

**Access detail required by local TB**

**Access detail required by global TB**

**Local TB miss**

**Global TB miss**

***FIG. 18D***

**U.S. Patent**

Apr. 20, 2004

Sheet 51 of 148

**US 6,725,356 B2**

1910

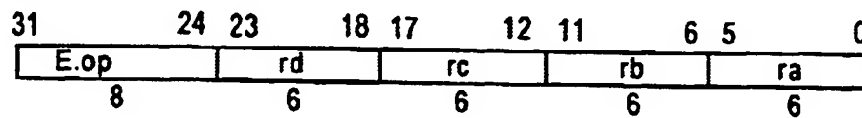
Operation codes

E.MUL.ADD.X	Ensemble multiply add extract
E.CON.X	Ensemble convolve extract

Format

E.op rd@rc,rb,ra

rd=gop(rd,rc,rb,ra)

**FIG. 19A**

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 52 of 148**

**US 6,725,356 B2**

1910

Figures 19B and 20B has blank fields: should be.

fsize	dpos	x	s	n	m	l	rnd	gssp
-------	------	---	---	---	---	---	-----	------

**FIG. 19B**

U.S. Patent

Apr. 20, 2004

Sheet 53 of 148

US 6,725,356 B2

1930

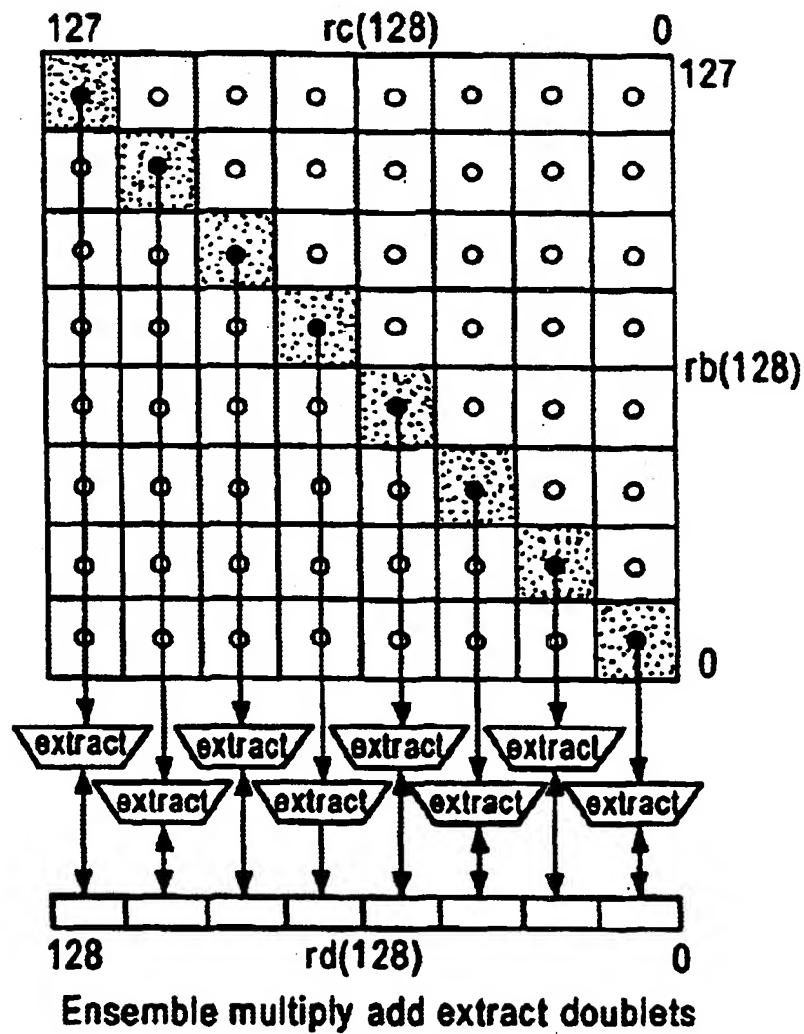
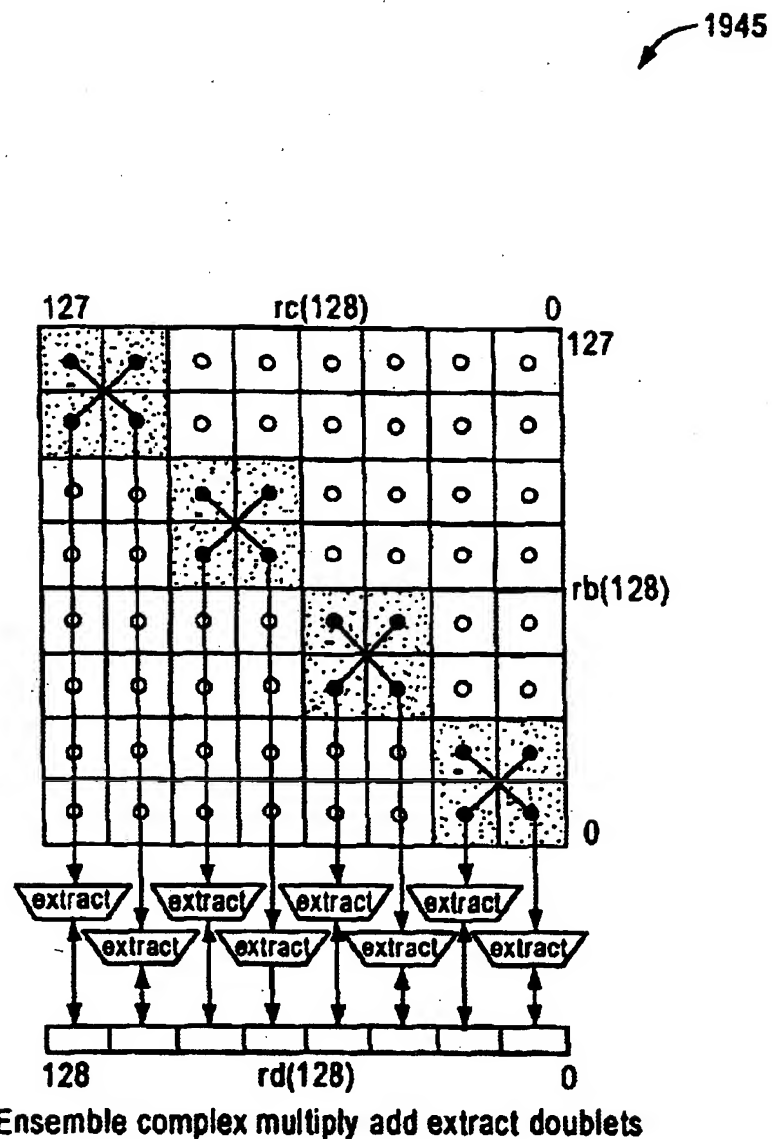


FIG. 19C





This ensemble-multiply-add-extract instructions (E.MUL.ADD.X), when the x bit is set, multiply the low-order 64 bits of each of the rc and rb registers and produce extended (double-size) results.

FIG. 19D

U.S. Patent

Apr. 20, 2004

Sheet 55 of 148

US 6,725,356 B2

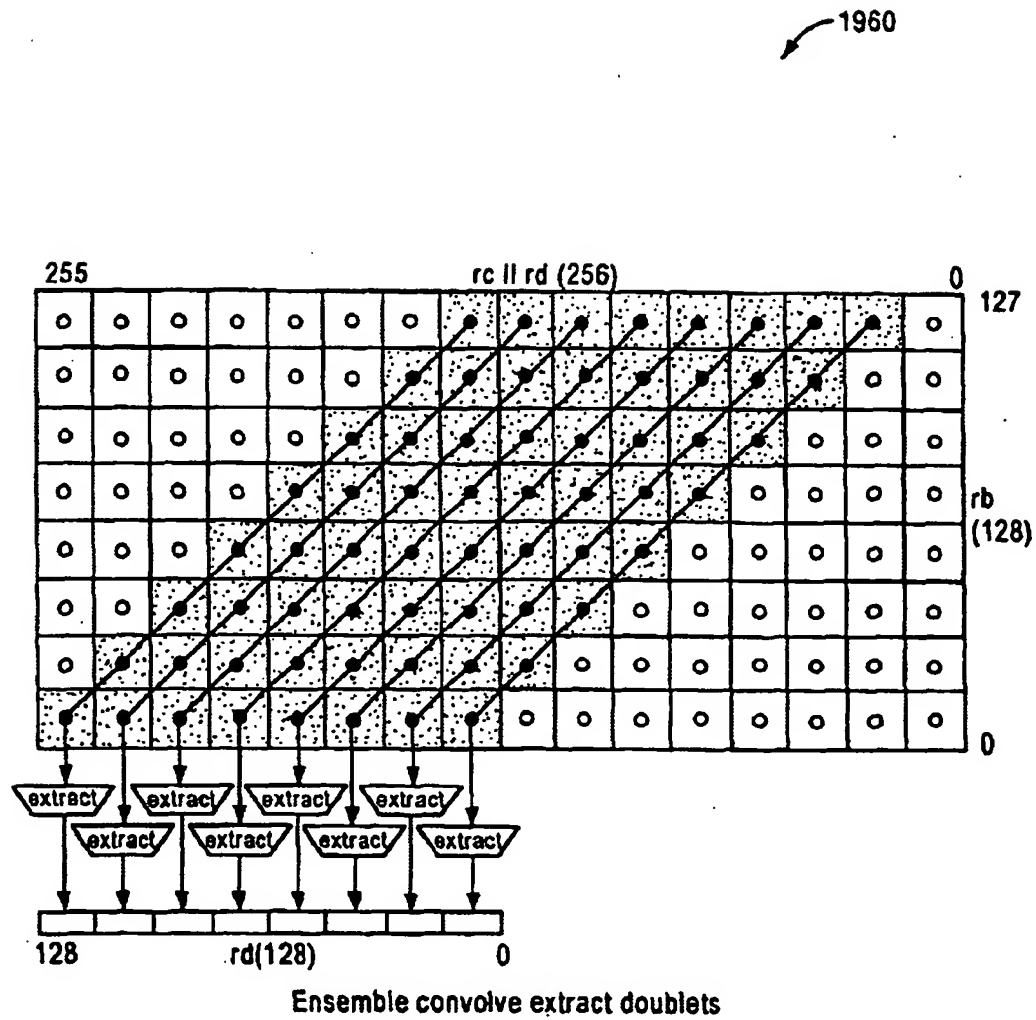


FIG. 19E

U.S. Patent

Apr. 20, 2004

Sheet 56 of 148

US 6,725,356 B2

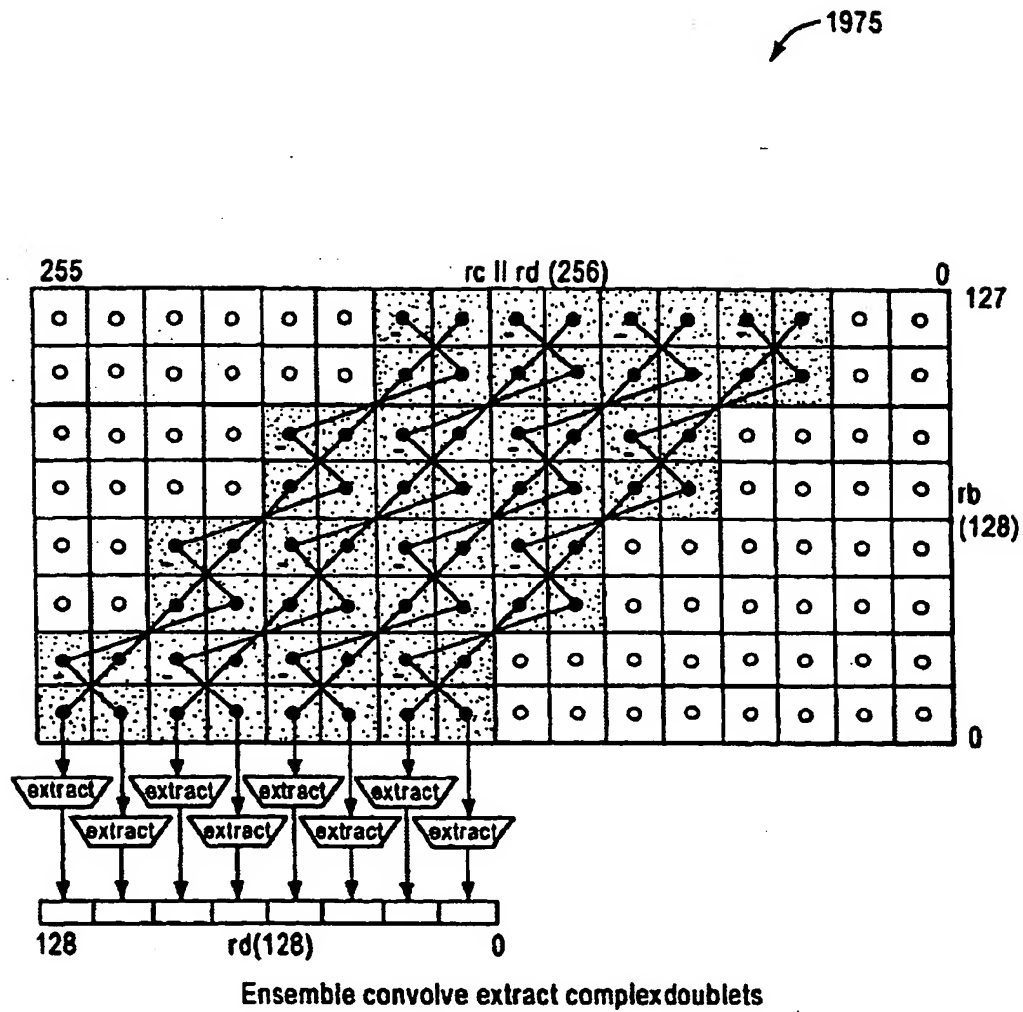


FIG. 19F

U.S. Patent

Apr. 20, 2004

Sheet 57 of 148

US 6,725,356 B2

Definition

def mul(size,h,vs,v,i,ws,w,j) as

mul ← ((vs & v<sub>size-1+j</sub>) h-size || v<sub>size-1+j</sub>) \* ((ws & w<sub>size-1+j</sub>) h-size || w<sub>size-1+j</sub>)

enddef

1990

def EnsembleExtractInplace(op,ra,rb,rc,rd) as

d ← RegRead(rd, 128)

c ← RegRead(rc, 128)

b ← RegRead(rb, 128)

case b<sub>8..0</sub> of

0..255:

sgsize ← 128

256..383:

sgsize ← 64

384..447:

sgsize ← 32

448..479:

sgsize ← 16

480..495:

sgsize ← 8

496..503:

sgsize ← 4

504..507:

sgsize ← 2

508..511:

sgsize ← 1

endcase

l ← a<sub>11</sub>m ← a<sub>12</sub>n ← a<sub>13</sub>signed ← a<sub>14</sub>x ← a<sub>15</sub>

case op of

E.CON.X:

if (sgsize &lt; 8) then

gsize ← 8

elseif (sgsize\*(n-1)\*(x+1) &gt; 128 then

gsize ← 128/(n-1)/(x+1)

else

gsize ← sgsz

endif

lgsize ← log(gsize)

wsz ← 128/(x+1)

FIG. 19G-1

U.S. Patent

Apr. 20, 2004

Sheet 58 of 148

US 6,725,356 B2

$vsiz \leftarrow 128$   
 $ds \leftarrow cs \leftarrow \text{signed}$   
 $bs \leftarrow \text{signed} \wedge m$   
 $zs \leftarrow \text{signed or } m \text{ or } n$   
 $zsize \leftarrow gsize \cdot (x+1)$   
 $h \leftarrow (2 \cdot gsize) + \log(vsiz) - lsize$   
 $spos \leftarrow (a_{8..0}) \text{ and } (2 \cdot gsize - 1)$

1990

E.MUL.ADD.X:

if( $sgsize < 9$ ) then  
      $gsize \leftarrow 8$   
 elseif ( $sgsize \cdot (n+1) \cdot (x+1) > 128$ ) then  
      $gsize \leftarrow 128 / (n+1) / (x+1)$   
 else  
      $gsize \leftarrow sgsiz$   
 endif  
 $ds \leftarrow \text{signed}$   
 $cs \leftarrow \text{signed} \wedge m$   
 $zs \leftarrow \text{signed or } m \text{ or } n$   
 $zsize \leftarrow gsize \cdot (x+1)$   
 $h \leftarrow (2 \cdot gsize) + n$   
 $spos \leftarrow (a_{8..0}) \text{ and } (2 \cdot gsize - 1)$   
 endcase  
 $dpos \leftarrow (0 \parallel a_{23..16}) \text{ and } (zsize - 1)$   
 $r \leftarrow spos$   
 $sfsiz \leftarrow (0 \parallel a_{31..24}) \text{ and } (zsize - 1)$   
 $tfsiz \leftarrow (sfsiz = 0) \text{ or } ((sfsiz + dpos) > zsize) ? zsize - dpos : sfsiz$   
 $fsize \leftarrow (tfsiz + spos > h) ? h - spos : tfsiz$   
 if ( $b_{10..9} = Z$ ) and not as then  
      $rnd \leftarrow F$   
 else  
      $rnd \leftarrow b_{10..9}$   
 endif

FIG. 19G-2

1990

```

for k ← 0 to wsize-zsize by zsize
  i ← k*gsize/zsize
  case op of
    E.CON.X:
      q[0] ← 0
      for j ← 0 to vsize-gsize by gsize
        if n then
          if (~) & j & gsize = 0 then
            q[j+gsize] ← q[j] + mul(gsize,h,ms,m,i+
              128-j,bs,b,j)
          else
            q[j+gsize] ← q[j] - mul(gsize,h,ms,m,i+
              128-j+2*gsize,bs,b,j)
          endif
        else
          q[j+gsize] ← q[j] + mul(gsize,h,ms,m,i+
            128-j,bs,b,j)
        endif
      endfor
      p ← q[vsize]
    E.MUL.ADD.X:
      di ← ((ds and dk+zsize-1)h-zsize-r || (dk+zsize-1..k) || 0')
      if n then
        if (i and gsize) = 0 then
          p ← mul(gsize,h,ds,d,i,cs,c,i)-
mul(gsize,h,ds,d,i+gsize,cs,c,i+gsize)+di
        else
          p ← mul(gsize,h,ds,d,i,cs,c,i+gsize)+mul(gsize,h,ds,d,i,cs,c,i+gsize)+di
        endif
      else
        p ← mul(gsize,h,ds,d,i,cs,c,i) + di
      endif
    endif
  endcase

```

FIG. 19G-3

U.S. Patent

Apr. 20, 2004

Sheet 60 of 148

US 6,725,356 B2

1990

```

case rnd of
  N:
     $s \leftarrow 0^{h-r} \parallel \sim p_r \parallel p_r^{r-1}$ 
  Z:
     $s \leftarrow 0^{h-r} \parallel p_{h-1}^r$ 
  F:
     $s \leftarrow 0^h$ 
  C:
     $s \leftarrow 0^{h-r} \parallel 1^r$ 
endcase
 $v \leftarrow ((zs \& p_{h-1}) \parallel p) + (0 \parallel s)$ 
if ( $v_{h..r+fsz} = (zs \& v_{r+fsz-1})^{h+1-r-fsz}$ ) or not (l and (op =
EXTRACT)) then
   $w \leftarrow (zs \& v_{r+fsz-1})^{zsize-fsz-dpos} \parallel v_{fsz-1+r..r} \parallel 0_{dpos}$ 
else
   $w \leftarrow (zs ? (v_h \parallel \sim v_h^{zsize-dpos-1}) : 1^{zsize-dpos}) \parallel 0_{dpos}$ 
endif
 $z_{size-1-k..k} \leftarrow w$ 
endfor
RegWrite(rd, 128, z)
enddef

```

FIG. 19G-4

**U.S. Patent**

Apr. 20, 2004

Sheet 61 of 148

**US 6,725,356 B2**

2010

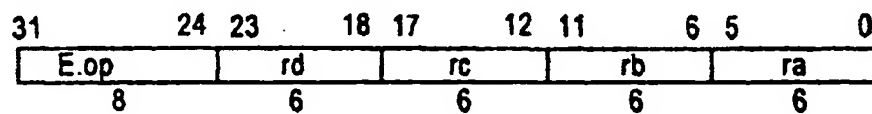
Operation codes

E.MUL.X	Ensemble multiply extract
E.EXTRACT	Ensemble extract
E.SCAL.ADD.X	Ensemble scale and extract

Format

E.op ra=rd,rc,rb

ra=eop(rd,rc,rb)

**FIG. 20A**



**U.S. Patent**

**Apr. 20, 2004**

**Sheet 62 of 148**

**US 6,725,356 B2**

2015

Figures 19B and 20B has blank fields: should be.

fsize	dpos	x	s	n	m	l	rnd	gssp
-------	------	---	---	---	---	---	-----	------

**FIG. 20B**

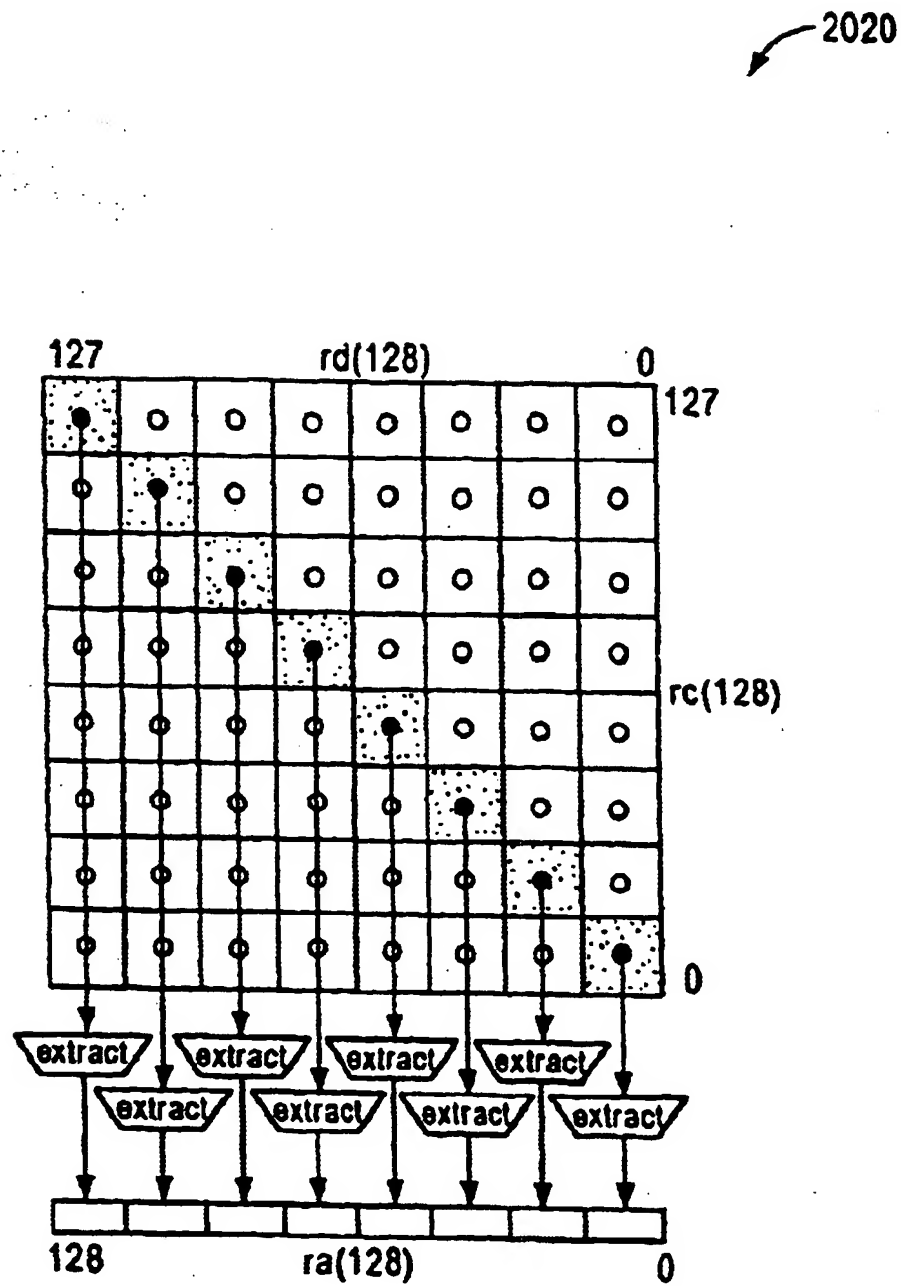
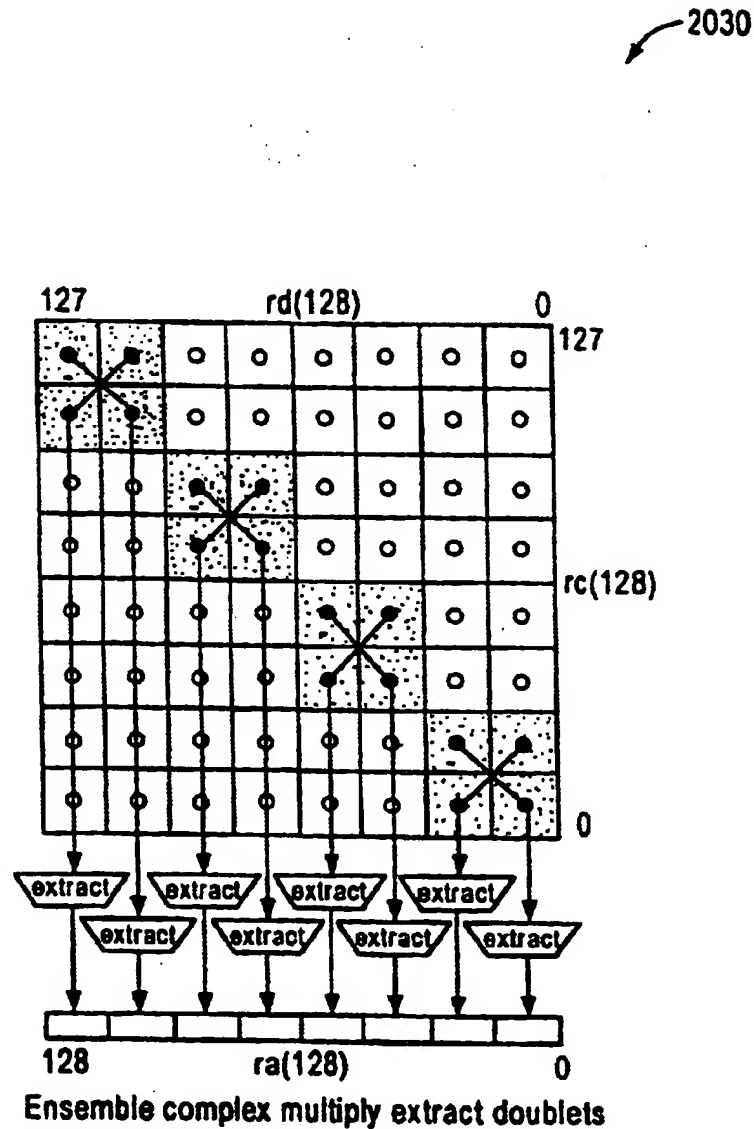


FIG. 20C



This ensemble-multiply-extract instructions (E.MUL.X), when the x bit is set, multiply the low-order 64 bits of each of the rc and rb registers and produce extended (double-size) results.

FIG. 20D

U.S. Patent

Apr. 20, 2004

Sheet 65 of 148

US 6,725,356 B2

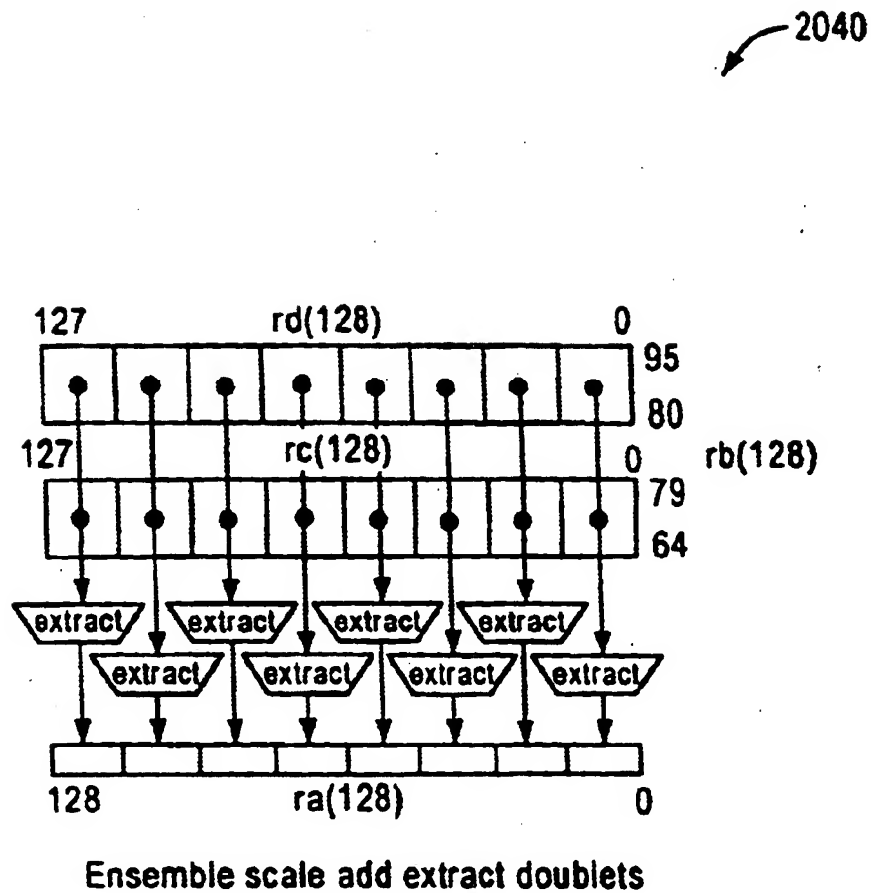
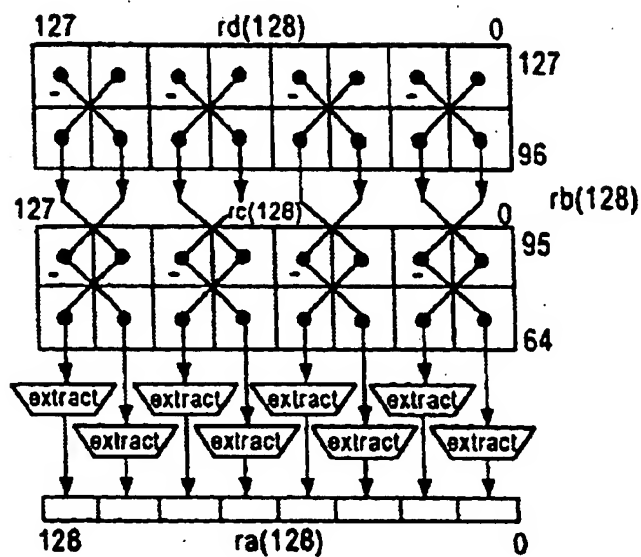


FIG. 20E



Ensemble complex scale add extract doublets

The ensemble-scale-add-extract instructions (E.SCLADD.X), when the x bit is set, multiply the low-order 64 bits of each of the rd and re registers by the rb register fields and produce extended (double-size) results.

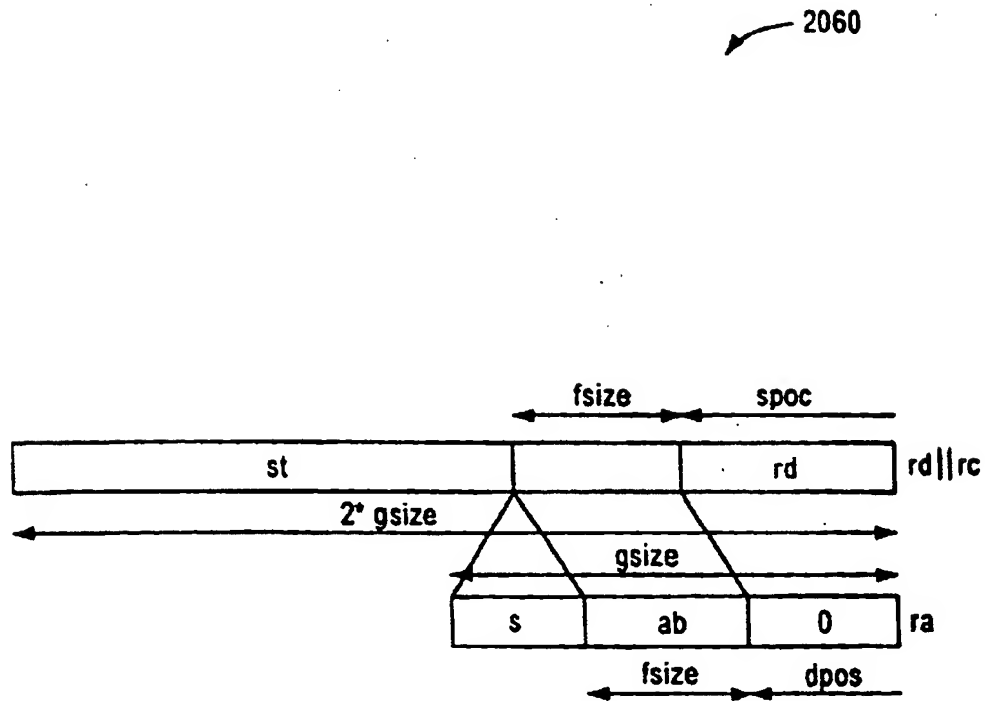
FIG. 20F

**U.S. Patent**

Apr. 20, 2004

Sheet 67 of 148

**US 6,725,356 B2**



Ensemble extract

**FIG. 20G**

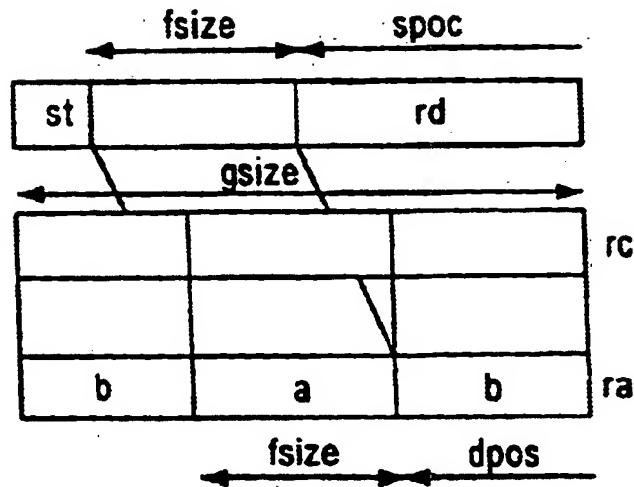
**U.S. Patent**

**Apr. 20, 2004**

**Sheet 68 of 148**

**US 6,725,356 B2**

2070



**Ensemble merge extract**

**FIG. 20H**

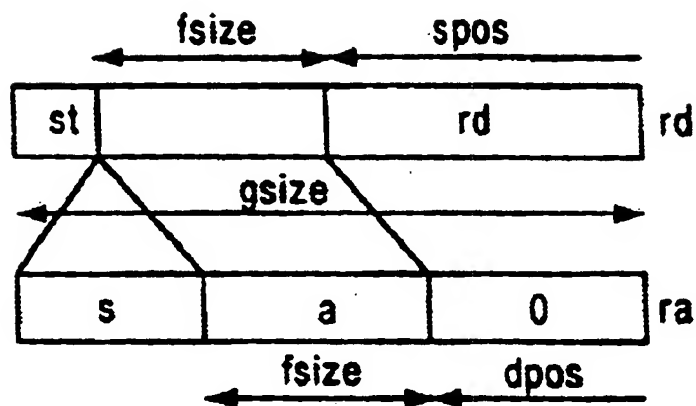
**U.S. Patent**

Apr. 20, 2004

Sheet 69 of 148

**US 6,725,356 B2**

2080



**Ensemble expand extract**

**FIG. 20I**



U.S. Patent

Apr. 20, 2004

Sheet 70 of 148

US 6,725,356 B2

Definition

def mul(size,h,vs,v,i,ws,w,j) as

mul ← ((vs&amp;vsize-1+i)h-size||vsize-1+i..i) \* ((ws&amp;ws-1+j)h-size||ws-1+j..j)

enddef

2090

def EnsembleExtract(op,ra,rb,rc,rd) as

d ← RegRead(rd, 128)

c ← RegRead(rc, 128)

b ← RegRead(rb, 128)

case ba..0 of

0..255:

sgsize ← 128

256..383:

sgsize ← 64

384..447:

sgsize ← 32

448..479:

sgsize ← 16

480..495:

sgsize ← 8

496..503:

sgsize ← 4

504..507:

sgsize ← 2

508..511:

sgsize ← 1

endcase

l ← b11

m ← b12

n ← b13

signed ← b14

x ← b15

case op of

E.EXTRACT:

gsize ← sgsz\*2(2-(m or x))

zsize ← sgsz

h ← gsize

as ← signed

spos ← (ba..0) and (gsz-1)

FIG. 20J-1

U.S. Patent

Apr. 20, 2004

Sheet 71 of 148

US 6,725,356 B2

2090

```

E.SCAL.ADD.X:
  if (sgsize < 8) then
    gsize ← 8
  elseif (sgsize*(n+1) > 32) then
    gsize ← 32/(n+1)
  else
    gsize ← sgsz
  endif
  ds ← cs ← signed
  bs ← signed ^ m
  as ← signed or m or n
  zsize ← gsize*(x+1)
  h ← (2*gsz) + 1 + n
  spos ← (b8..0) and (2*gsz-1)
E.MUL.X:
  if (sgsize < 8) then
    gsize ← 8
  elseif (sgsize*(n+1)*(x+1) > 128) then
    gsize ← 128/(n+1)/(x+1)
  else
    gsize ← sgsz
  endif
  ds ← signed
  cs ← signed ^ m
  as ← signed or m or n
  zsize ← gsize*(x+1)
  h ← (2*gsz) + n
  spos ← (b8..0) and (2*gsz-1)
endcase
dpos ← (0|| b23..16) and (zsize-1)
r ← spos
sfsz ← (0|| b31..24) and (zsize-1)
tfsz ← (sfsz = 0) or ((sfsz+dpos) > zsize) ? zsize-dpos : sfsz
fsz ← (tfsz + spos > h) ? h - spos : tfsz
if (b10..9=Z) and not as then
  rnd ← F
else
  rnd ← b
endif

```

FIG. 20J-2

U.S. Patent

Apr. 20, 2004

Sheet 72 of 148

US 6,725,356 B2

2090

```

for j ← 0 to 128-zsize by zsize
  i ← j*gszsize/zsize
  case op of
    E.EXTRACT:
      if m or x then
        p ← dgszsize+i-1..i
      else
        p ← (d||c)gszsize+i-1..i
      endif
    E.MUL.X:
      if n then
        if (i and gszsize) = 0 then
          p ← mul(gszsize,h,ds,d,i,cs,c,i)-
mul(gszsize,h,ds,d,i+gszsize,cs,c,i+gszsize)
        else
          p ←
mul(gszsize,h,ds,d,i,cs,c,i+gszsize)+mul(gszsize,h,ds,d,i,cs,c,i+gszsize)
        endif
      else
        p ← mul(gszsize,h,ds,d,i,cs,c,i)
      endif
    E.SCAL.ADD.X:
      if n then
        if (i and gszsize) = 0 then
          p ← mul(gszsize,h,ds,d,i,bs,b,64+2*gszsize)
            + mul(gszsize,h,cs,c,i,bs,b,64)
            - mul(gszsize,h,ds,d,i+gszsize,bs,b,64+3*gszsize)
            - mul(gszsize,h,cs,c,i+gszsize,bs,b,64+gszsize)
        else
          p ← mul(gszsize,h,ds,d,i,bs,b,64+3*gszsize)
            + mul(gszsize,h,cs,c,i,bs,b,64+gszsize)
            + mul(gszsize,h,ds,d,i+gszsize,bs,b,64+2*gszsize)
            + mul(gszsize,h,cs,c,i+gszsize,bs,b,64)
        endif
      else
        p ← mul(gszsize,h,ds,d,i,bs,b,64+gszsize) + mul(gszsize
,h,cs,c,i,bs,b,64)
      endif
    endif
  endcase

```

FIG. 20J-3

U.S. Patent

Apr. 20, 2004

Sheet 73 of 148

US 6,725,356 B2

```

case rnd of
  N:
     $s \leftarrow 0^{h-r} \parallel \sim p_r \parallel p_r^{f-1}$ 
  Z:
     $s \leftarrow 0^{h-r} \parallel p_{h-1}^r$ 
  F:
     $s \leftarrow 0^h$ 
  C:
     $s \leftarrow 0^{h-r} \parallel 1^r$ 
endcase
v  $\leftarrow ((as \& p_{h-1}) \parallel p) + (0 \parallel s)$ 
if  $(v_{h..r+fsize} = (as \& v_{r+fsize-1})^{h+1-r-fsize})$  or not (1 and (op =
  E.EXTRACT)) then
  w  $\leftarrow (as \& v_{r+fsize-1})^{zsize-fsize-dpos} \parallel v_{fsize-1+r..r} \parallel 0^{dpos}$ 
else
  w  $\leftarrow (s ? (v_h \parallel \sim v_h^{zsize-dpos-1}) : 1^{zsize-dpos}) \parallel 0^{dpos}$ 
endif
if m and (op = E.EXTRACT) then
   $z_{zsize-1+j..j} \leftarrow c_{asize-1+j..dpos+fsize+j} \parallel w_{dpos+fsize-1..dpos} \parallel$ 
     $c_{dpos-1+j..j}$ 
else
   $z_{zsize-1+j..j} \leftarrow w$ 
endif
endfor
RegWrite(ra, 128, z)
enddef

```

2090

FIG. 20J-4

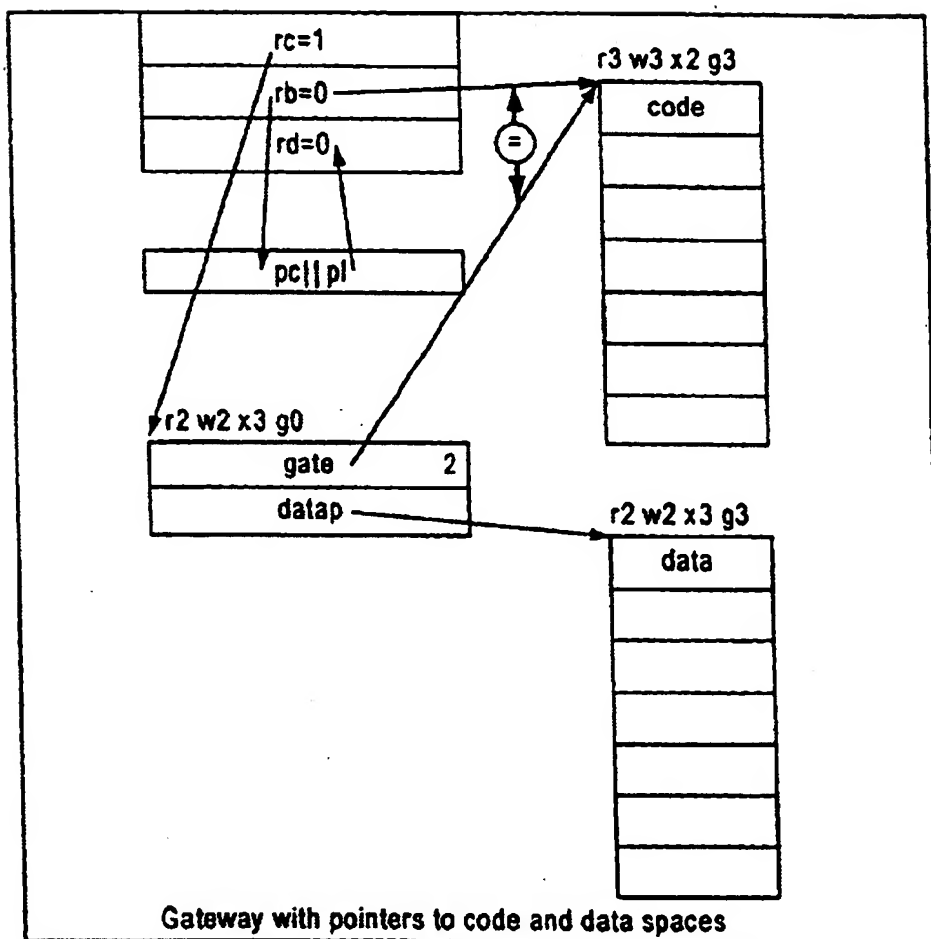
**U.S. Patent**

Apr. 20, 2004

Sheet 74 of 148

**US 6,725,356 B2**

2110



**FIG. 21A**

**U.S. Patent**

Apr. 20, 2004

Sheet 75 of 148

**US 6,725,356 B2**

2130

Typical dynamic-linked, inter-gateway calling sequence:

caller:

caller	AA.DDI	sp@-size	// allocate caller stack frame
	S.I.64.A	lp,sp,off	
	S.I.64.A	dp,sp,off	
	...		
	L.I.64.A	lp=dp,off	// load lp
	L.I.64.A	dp=dp,off	// load dp
	B.GATE		
	L.I.64.A	dp,sp,off	
	...(code using dp)		
	L.I.64.A	lp=sp,off	// restore original lp register
	A.ADDI	sp=size	// deallocate caller stack frame
	B	lp	// return

callee (non-leaf):

callee:	L.I.64.A	dp=dp,off	// load dp with data pointer
	S.I.64.A	sp,dp,off	
	L.I.64.A	sp=dp,off	// new stack pointer
	S.I.64.A	lp,sp,off	
	S.I.64.A	dp,sp,off	
	...(using dp)		
	L.I.64.A	dp,sp,off	
	...(code using dp)		
	L.I.64.A	lp=sp,off	// restore original lp register
	L.I.64.A	sp=sp,off	// restore original sp register
	B.DOWN	lp	

callee (leaf, no stack):

callee:	...(using dp)	
	B.DOWN	lp

**FIG. 21B**

**U.S. Patent**

Apr. 20, 2004

Sheet 76 of 148

**US 6,725,356 B2**

2160

Operation codes

B.GATE	Branch gateway
--------	----------------

Equivalencies

B.GATE	← B.GATE 0
--------	------------

Format

B.GATE rb

bgate(rb)

31	24	23	18	17	12	11	6	5	0
B.MINOR		0	1		rb		B.GATE		
8		6	6		6		6		

**FIG. 21C**

2170

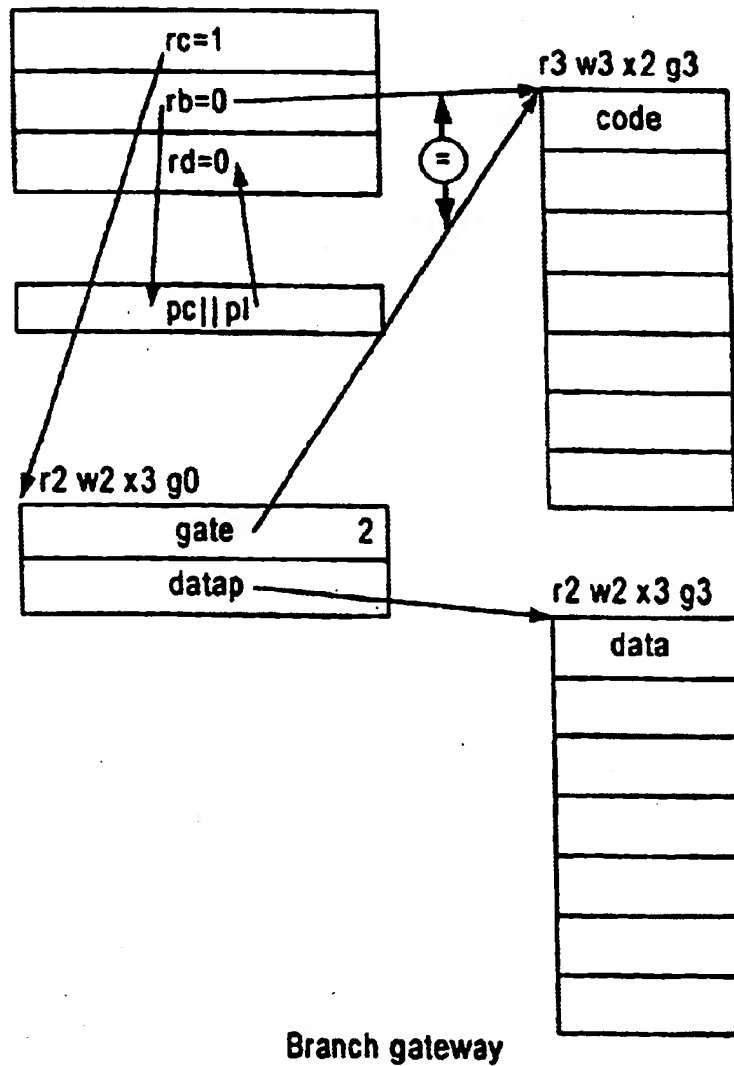


FIG. 21D




U.S. Patent

Apr. 20, 2004

Sheet 78 of 148

US 6,725,356 B2

 2190
Definition

```

def BranchGateway(rd,rc,rb) as
  c ← RegRead(rc, 64)
  b ← RegRead(rb, 64)
  if (rd ≠ 0) or (rc ≠ 1) then
    raise ReservedInstruction
  endif
  if c2..0 ≠ 0 then
    raise AccessDisallowedByVirtualAddress
  endif
  d ← ProgramCounter63..2+1 || PrivilegeLevel
  if PrivilegeLevel < b1..0 then
    m ← LoadMemoryG(c,c,64,L)
    if b ≠ m then
      raise GatewayDisallowed
    endif
    PrivilegeLevel ← b1..0
  endif
  ProgramCounter ← b63..2 || 02
  RegWrite(rd, 64, d)
  raise TakenBranch
enddef

```

FIG. 21E

**U.S. Patent**

**Apr. 20, 2004**

**Sheet 79 of 148**

**US 6,725,356 B2**

**2199**



**Exceptions**

**Reserved Instruction**

**Gateway disallowed**

**Access disallowed by virtual address**

**Access disallowed by tag**

**Access disallowed by global TB**

**Access disallowed by local TB**

**Access detail required by tag**

**Access detail required by local TB**

**Access detail required by global TB**

**Local TB miss**

**Global TB miss**

**FIG. 21F**

**U.S. Patent**

Apr. 20, 2004

Sheet 80 of 148

**US 6,725,356 B2**

2210

Operation codes

E.SCALADD.F.16	Ensemble scale add floating-point half
E.SCALADD.F.32	Ensemble scale add floating-point single
E.SCALADD.F.64	Ensemble scale add floating-point double

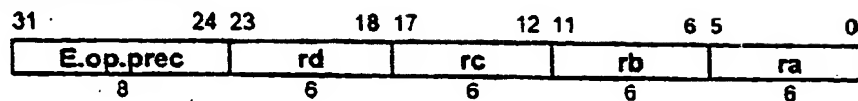
Selection

class	op	prec		
scale add	E.SCALADD.F	16	32	64

Format

E.op.prec ra=rd,rc,rb

ra=eopprec(rd,rc,rb)


**FIG. 22A**

U.S. Patent

Apr. 20, 2004

Sheet 81 of 148

US 6,725,356 B2

 2230
Definition

```

def EnsembleFloatingPointTernary(op, prec, rd, rc, rb, ra) as
  d ← RegRead(rd, 128)
  c ← RegRead(rc, 128)
  b ← RegRead(rb, 128)
  for i ← 0 to 128-prec by prec
    di ← F(prec, di+prec-1..i)
    ci ← F(prec, ci+prec-1..i)
    ai ← fadd(fmul(di, F(prec, bprec-1..0)), fmul(ci, F(prec, b2+prec-1..prec)))
    ai+prec-1..i ← PackF(prec, ai, none)
  endfor
  RegWrite(ra, 128, a)
enddef

```

FIG. 22B

U.S. Patent

Apr. 20, 2004

Sheet 82 of 148

US 6,725,356 B2

2310

Operation codes

G.BOOLEAN	Group boolean
-----------	---------------

Selection

operation	function (binary)	function (decimal)
d	11110000	240
c	11001100	204
b	10101010	178
d&c&b	10000000	128
(d&c) b	11101010	234
d c b	11111110	254
d?c:b	11001010	202
d^c^b	10010110	150
~d^c^b	01101001	105
0	00000000	0

Format

G.BOOLEAN rd@trc,rb,f

rd=gbooleani(rd,rc,rb,f)

31	25	24	23	18	17	12	11	6	5	0
G.BOOLEAN	ih	rd	rc	rb	il					
7	1	6	6	6	6					

FIG. 23A

U.S. Patent

Apr. 20, 2004

Sheet 83 of 148

US 6,725,356 B2

2320

```

if f6=f5 then
  if f2=f1 then
    if f2 then
      rc ← max(trc, trb)
      rb ← min(trc, trb)
    else
      rc ← min(trc, trb)
      rb ← max(trc, trb)
    endif
    ih ← 0
    il ← 0 || f6 || f7 || f4 || f3 || f0
  else
    if f2 then
      rc ← trb
      rb ← trc
    else
      rc ← trc
      rb ← trb
    endif
    ih ← 0
    il ← 1 || f6 || f7 || f4 || f3 || f0
  endif
else
  ih ← 1
  if f6 then
    rc ← trb
    rb ← trc
    il ← f1 || f2 || f7 || f4 || f3 || f0
  else
    rc ← trc
    rb ← trb
    il ← f2 || f1 || f7 || f4 || f3 || f0
  endif
endif
endif

```


FIG. 23B

U.S. Patent

Apr. 20, 2004

Sheet 84 of 148

US 6,725,356 B2

 2330
Definition

```

def GroupBoolean (ih,rd,rc,rb,il)
  d ← RegRead(rd, 128)
  c ← RegRead(rc, 128)
  b ← RegRead(rb, 128)
  if ih=0 then
    if il5=0 then
      f ← il3 || il4 || il4 || il2 || il1 || (rc>rb)2 || il0
    else
      f ← il3 || il4 || il4 || il2 || il1 || 0 || 1 || il0
    endif
  else
    f ← il3 || 0 || 1 || il2 || il1 || il5 || il4 || il0
  endif
  for i ← 0 to 127 by size
    ai ← f(di||ci||bi)
  endfor
  RegWrite(rd, 128, a)
enddef

```

FIG. 23C

**U.S. Patent**

Apr. 20, 2004

Sheet 85 of 148

**US 6,725,356 B2**

2410

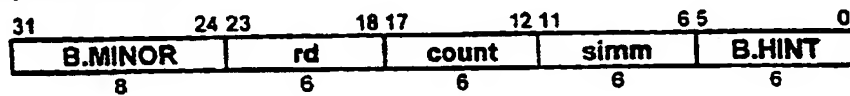
Operation codes

<b>B.HINT</b>	<b>Branch Hint</b>
---------------	--------------------

Format

**B.HINT**      **badd,count,rd**

**bhint(badd,count,rd)**



**simm** ← **badd-pc-4**

**FIG. 24A**